Stefania Loredana Nita
Marius Iulian Mihailescu

Advances to Homomorphic and Searchable Encryption

Springer

Advances to Homomorphic and Searchable Encryption

Stefania Loredana Nita · Marius Iulian Mihailescu

Advances to Homomorphic and Searchable Encryption

Springer

Stefania Loredana Nita
"FERDINAND I" Military Technical Academy
Bucharest, Romania

Marius Iulian Mihailescu
"SPIRU HARET" University
Bucharest, Romania

ISBN 978-3-031-43213-2 ISBN 978-3-031-43214-9 (eBook)
https://doi.org/10.1007/978-3-031-43214-9

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Paper in this product is recyclable.

To our families

Preface

In the ever-evolving digital landscape of the twenty-first century, data security and privacy have emerged as paramount concerns. With the widespread adoption of cloud computing and the proliferation of sensitive information being stored and processed remotely, traditional security measures have proven inadequate to protect the confidentiality of our data. In response, the field of cryptography has witnessed remarkable advancements, paving the way for innovative solutions that offer both confidentiality and functionality.

This book, "Advances to Homomorphic and Searchable Encryption," delves into two of the most groundbreaking cryptographic techniques of our time: Searchable Encryption (SE) and Homomorphic Encryption (HE). These techniques have revolutionized the way we interact with encrypted data, enabling powerful operations without compromising privacy. Searchable Encryption is a fascinating concept that allows searching through encrypted data without the need for decryption. It offers a bridge between data security and data usability, enabling users to securely outsource their data to third-party services while preserving their privacy. Over the past decade, significant progress has been made in the development of efficient and secure searchable encryption schemes, which have found numerous applications in areas such as health care, e-commerce, and cloud computing.

Homomorphic Encryption, on the other hand, represents a paradigm shift in cryptography by enabling computations to be performed directly on encrypted data, yielding encrypted results that can be decrypted to obtain the desired outcomes. This breakthrough technology has the potential to transform the way we process sensitive information, allowing computations to be outsourced to untrusted parties while ensuring the confidentiality of the data. The advent of practical homomorphic encryption schemes has opened up new avenues for secure computation, making it a topic of great interest and research in academia and industry alike.

This book brings together a comprehensive collection of contributions from leading experts and researchers in the field of searchable encryption and homomorphic encryption. It serves as a valuable resource for academics, practitioners, and students seeking to explore the latest advancements, emerging trends, and real-world applications of these two transformative cryptographic techniques. Each chapter

delves into specific aspects, including theoretical foundations, novel schemes, practical implementations, and potential challenges, providing a holistic view of the subject matter.

While the challenges and complexities of designing efficient and secure searchable encryption and homomorphic encryption schemes are formidable, the potential benefits they offer in terms of data security and privacy are unparalleled. As we navigate a world increasingly reliant on the secure processing of encrypted data, it is our hope that this book will inspire further innovation, collaboration, and exploration in the exciting field of searchable encryption and homomorphic encryption.

Bucharest, Romania
July 2023

Stefania Loredana Nita
Marius Iulian Mihailescu

Acknowledgements We would like to thank our families for their understanding and our colleagues for their valuable comments. We extend our heartfelt gratitude to all the researchers, professors, and experts in the field, whose dedication and expertise have made this book possible by approaching current subjects in cryptography and cybersecurity. Their contributions will undoubtedly shape the future of cryptography and empower individuals and organizations to embrace the full potential of secure data processing.

Contents

Acronyms

ABE	Attribute-based Encryption
ACGD	Approximate-Greatest Common Divisor
AI	Artificial Intelligence
AR	Augmented Reality
AWS	Amazon Web Services
BDDP	Bounded Distance Decoding Problem
BFV	Brakerski-Fan-Vercauteren cryptosystem
BGV	Brakerski-Gentry-Vaikuntanathan cryptosystem
BV	Brakerski-Vaikuntanathan cryptosystem
CAGR	Compound Annual Growth Rate
CGGI	Chillotti-Gama-Georgieva-Izabachène cryptosystem (also known as TFHE)
CKKS	Cheon-Kim-Kim-Song cryptosystem
DBDH	Decisional Bilinear Diffie-Hellman
DOFH	Decisional Oracle Diffie-Hellman
FHE	Fully Homomorphic Encryption
GPU	Graphics Processing Unit
HE	Homomorphic Encryption
IBE	Identity-based Encryption
ICT	Information and Communications Technology
IKGA	Inside Keyword Guessing Attacks
IoT	Internet of Things
ISO/IEC	International Organization for Standardization/International Electrotechnical Commission
ISVP	Ideal-Shortest Vector Problem
LWE	Learning with Errors
ML	Machine Learning
MPC	Multiparty Computation
NIST	National Institute of Standards and Technology
PHE	Partial Homomorphic Encryption
PKC	Public-Key Cryptography

PPT	Probabilistic Polynomial-Time
PQC	Post-Quantum Cryptography
SE	Searchable Encryption
SSSP	Sparse Subset Sum Problem
SWHE	Somewhat Homomorphic Encryption
TFHE	Fast Fully Homomorphic Encryption over the Torus (also known as CGGI)
VR	Virtual Reality

Chapter 1
Introduction

1.1 Motivation

Information and communications technology (ICT) has significantly increased its potential over the past decade, as the speed of computations and communications have increased. In the sphere of information and communications technology, cloud computing plays a significant role and has made significant contributions, becoming one of the most prominent technologies today. Amazon, with Amazon Web Services (AWS), Google, with Google Cloud, Microsoft, with Microsoft Azure, Massive Grid, Oracle, with Oracle Cloud, IBM, etc., are examples of well-known global cloud service providers. Cloud computing is a technology that is utilized by both individual consumers and businesses, offering various service models. From the perspective of an individual user, storage capacity may be one of the cloud's most essential features. The ability to edit documents online, such as with Google Docs, is another advantage of utilizing cloud services. This eliminates the need for users to transport multiple versions of the same document on their various devices. With cloud services, the user can access and edit his or her document from any device, at any time, via a web browser. Because the user accesses the cloud via the Internet, the Internet connection is very important in this process. However, some providers offer the option of working offline, therefore the user can work offline and the synchronization occurs when there is an Internet connection. Businesses benefit even more from cloud computing since they may use cloud services to externalize their whole IT infrastructure. Cloud computing is a technology that works in tandem with big data technology, which consists of large dynamic volumes of structured (such as traditional databases) and unstructured (such as files and documents, video files, audio files, image files, and so on) data that is a gold mine for statistics, reports, and any data analysis process. A forecast by Gartner [1] shows that public cloud services will grow 21.7% to total $597.3 billion in 2023, compared to $491 billion in 2022.

Furthermore, we stand witness to an explosion of data generated by the rapid advancements in technologies like artificial intelligence (AI), machine learning (ML),

S. L. Nita and M. I. Mihailescu, *Advances to Homomorphic and Searchable Encryption*,
https://doi.org/10.1007/978-3-031-43214-9_1

Internet of Things (IoT), cloud computing, big data, smart cities, etc. These technologies, while remarkably transforming our world and the way we live, also intensify the challenges related to data privacy and security. Machine learning and artificial intelligence, for instance, depend heavily on large amounts of data to train models and generate insights. However, the sensitive nature of the data poses serious concerns about privacy and security. Similarly, IoT devices and smart city infrastructures, with their interconnections and constant data exchanges, are important sources for data breaches. Big data technologies handle enormous data volumes, which increases the scope and impact of any potential data leak. Cloud computing presents another significant concern. While it offers scalable, cost-effective solutions for data storage and processing, it also introduces risks. The data stored on the cloud is often out of the direct control of the owners, posing significant privacy challenges. A Research and Markets report [2] forecasts that "*global AI training dataset market size is expected to reach USD 8.61 billion by 2030 and expand at a CAGR of 22.1% from 2023 to 2030*".

Considering how much data is stored in the cloud, it is natural to wonder how this data is protected. However, this topic applies to all Internet-based data and information. The cloud environment cannot be incorporated with a company's existing security infrastructure due to the fact that both environments employ distinct security procedures. Many businesses believe that security is solely the responsibility of the cloud service provider, but in reality, the business itself plays an essential role in securing certain components of its environment. The cloud provider is responsible for maintaining access to and from the cloud and securing elements such as storage and infrastructure, while the company is responsible for securing the data stored in the cloud, configuring the network and firewall properly, applying server-side encryption, etc. Another factor that must be considered is the type of service models employed by the organization, as each of them can obtain access to the organization in a unique manner, which can result in security vulnerabilities. However, the majority of cloud computing's security issues are caused by the cloud's poor administration of authentication and authorization. The loss or disclosure of the data is the result of security breaches, excessive resource utilization by the services, cyberattacks, etc. Cloud computing is a continually developing and evolving technology. This also applies to cloud security concerns. Due to its complex infrastructure, the cloud may inherit the security flaws of its components; therefore, everything should be protected, from the cloud environment to the hosted public services. An attack can be launched from any Internet-connected or cloud-services-using device. Cyber attacks have been ranked as the fifth greatest danger for the year 2020 and have become the norm in both the public and private sectors. This dangerous industry will continue to expand in 2023, with IoT cyber attacks expected to double by 2025. In addition, according to the World Economic Forum's 2020 Global Risk Report, the rate of detection (or prosecution) in the United States is as low as 0.5% [3]. Another report by IBM [4] reveals that the average total cost associated with a ransomware breach stands at $4.62 million, slightly surpassing the $4.24 million average cost of a general data breach. There has been a notable increase of 10.3% in the per capita cost of a data breach from 2020 to 2021. The healthcare sector witnessed a

significant surge in average total breach costs, escalating from $7.13 million in 2020 to $9.23 million in 2021—a rise of nearly 29.5%. In 2021, lost business opportunities contributed the most to breach costs, amounting to an average total cost of $1.59 million. A breach with a lifecycle exceeding 200 days bore an average cost of $4.87 million. Remarkably, 39% of the costs were experienced more than a year post the data breach incident. In a geographical breakdown, the United States topped the list in 2021 with the highest average total cost for a data breach, standing at $9.05 million. These statistics underscore the escalating financial impact of data breaches, emphasizing the need for robust security measures and timely incident response.

Taking into account all of these factors, it is not sufficient to merely use a secure communication channel to safeguard the data; additional measures must be taken. Homomorphic encryption (HE) and searchable encryption (SE) are two cryptographic techniques that can be used to resolve these obstacles. Homomorphic encryption has captivated the research community's attention over the past decade, particularly fully homomorphic encryption, regarded as "*the holy grail of cryptography*" [5]. While *Fully* Homomorphic Encryption (FHE) permits any computation on encrypted data, homomorphic encryption permits only some computations or any computations, but only for a limited number of repetitions. Searchable encryption is a subtype of fully homomorphic encryption that enables direct search operations on encrypted data. Consequently, both techniques have great potential in addressing contemporary security issues. Homomorphic Encryption and searchable encryption offer effective solutions to these challenges. HE allows computations to be performed on encrypted data without needing to decrypt it first, crucial for privacy-preserving computations. This is particularly essential for sectors like healthcare, finance, and technologies like machine learning, where sensitive data needs to be processed without risking exposure. SE, in contrast, allows selective search through encrypted data without decryption, ensuring data confidentiality while maintaining its utility. This is critically relevant in cloud storage, where users often need to retrieve specific data parts without exposing the entirety of their data to potential breaches.

The necessity for such advanced encryption techniques is escalating in our increasingly data-reliant society. As we face stricter data protection regulations and an impending era of quantum computing, which may render traditional encryption methods ineffective, understanding and implementing HE and SE becomes not just an academic pursuit but an essential requirement.

1.2 Book Structure

The aim of the book is to present the current state of the literature on the fields of homomorphic and searchable encryption, from theoretical and practical points of view. The book provides a comprehensive overview of the most recent research and advancements in these fields, highlighting new approaches to building more secure and efficient encryption systems, such as exploring the use of new algorithms, protocols, or hardware implementations to improve the performance and reliability of

searchable and homomorphic encryption. In addition, the book provides case studies and real-world examples that illustrate how searchable and homomorphic encryption can be applied in novel ways to solve complex problems, such as examining the potential applications of these encryption techniques in emerging fields such as the Internet of Things (IoT), Blockchain, Machine Learning (ML) and Artificial Intelligence (AI). The book has the following structure:

1. This chapter (*Introduction*) presents the context and motivation for the present work. Also, explains for short what are homomorphic and searchable encryption and why are these important.
2. Chapter 2 (*Background and Preliminaries*) aims to familiarize the reader with the mathematical concepts and notations used in this book and cryptography terminology and notations, as the topics of the book are advanced topics in cryptography. Also, the chapter presents which are the challenges for the security of cloud computing and big data and why searchable and homomorphic encryption may resolve (some of) them.
3. Chapter 3 (*Homomorphic Encryption*) presents the beginnings of homomorphic encryption, definitions, and types. Also, an entire section is focused on fully homomorphic encryption, the most interesting and promising type of homomorphic encryption. Further, the chapter presents recent advancements in homomorphic encryption discussing the advantages, disadvantages, performance, security requirements and presents case studies and practical applications in different domains (for example, Internet-of-Things, Machine Learning, Artificial Intelligence, etc.) Lastly, the chapter presents the challenges and research directions for homomorphic encryption.
4. Chapter 4 (*Searchable Encryption*) presents the beginnings of searchable encryption, definitions, and types. Also, one section is focused on the components of searchable encryption schemes, presenting the architectures, security requirements, and search operation types. Further, the chapter presents recent advancements in searchable encryption discussing the advantages, disadvantages, performance, security requirements, and presents case studies and practical applications in different domains (for example, Internet-of-Things, Machine Learning, Artificial Intelligence, etc.). Lastly, the chapter presents the challenges and research directions for searchable encryption.

References

1. Gartner Forecasts Worldwide Public Cloud End-User Spending to Reach Nearly $600 Billion in 2023. Gartner (2023). https://www.gartner.com/en/newsroom/press-releases/2023-04-19-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-nearly-600-billion-in-2023
2. AI Training Dataset Market Size, Share and Trends Analysis Report, by Type (Text, Image/Video, Audio), by Vertical (IT, Automotive, Government, Healthcare, BFSI), by Regions, and

Segment Forecasts, 2023–2030 (2023). https://www.researchandmarkets.com/reports/5440500/ai-training-dataset-market-size-share-and
3. 2023 Must-Know Cyber Attack Statistics and Trends | Embroker (2019). https://www.embroker.com/blog/cyber-attack-statistics
4. Cost of a Data Breach Report 2022 (2022). https://www.ibm.com/downloads/cas/3R8N1DZJ
5. Wu, D.J.: Fully homomorphic encryption: cryptography's holy grail. XRDS **21**, 24–29 (2015)

Chapter 2
Background and Preliminaries

2.1 Mathematical Background and Notations

Mathematics is the foundation of cryptography, providing the necessary tools and ideas that support cryptographic systems. Cryptographic algorithms' strength and security are basically based on mathematical ideas and structures such as number theory, algebra, and probability theory. These mathematical fields provide a diverse set of issues that are simple to describe yet difficult to answer computationally, making them excellent for building cryptographic systems. Many encryption systems rely on computationally expensive operations such as factoring huge integers or solving discrete logarithm issues. Furthermore, mathematical notions like as groups, rings, and fields are utilized to describe operations in a variety of cryptographic systems. Probability theory is important in generating random keys and determining the probability of successful assaults. As a result, a thorough grasp of fundamental mathematical concepts is required for designing, implementing, and analyzing safe cryptographic systems. This section will go through the important mathematical principles and notations utilized throughout this book, laying the groundwork for the future study of cryptography.

The definitions and descriptions from this section are according to [1–4].

Sets. The following standard notations for well-known sets will be used throughout this document: $\mathbb{N}$ represents the set of all natural numbers, $\mathbb{Z}$ denotes the set of all integer numbers, $\mathbb{R}$ signifies the set of all real numbers, and $\mathbb{C}$ stands for the set of all complex numbers. The notation $\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z} = \{0, \ldots, n-1\}, n \geq 2$ is used to represent the set of integers modulo the integer n. For a prime integer value p, the field is represented as $\mathbb{F}_p = \mathbb{Z}_p$. Lastly, $\mathcal{P}(S)$ is used to denote the set of all subsets of S.

Algebraic structures. An algebraic structure is formed up of the following elements: a set A with at least one element, a collection of operations defined on A, and a set of specifications that these operations must meet.

S. L. Nita and M. I. Mihailescu, *Advances to Homomorphic and Searchable Encryption*,
https://doi.org/10.1007/978-3-031-43214-9_2

Group. The algebraic structures most frequently utilized in this book are abelian groups and fields. Given a non-empty set G and a binary operation "+" defined on G, the structure $(G, +)$ is an *abelian group* if it satisfies the following conditions:

- The operation "+" is well-defined on G, meaning that for any $g_1, g_2 \in G$, the result $+(g_1, g_2)$ is also in G. This property is inherently satisfied as "+" is a binary operation.
- The operation "+" is associative. This means that for any $g_1, g_2, g_3 \in G$, the equation $+(+(g_1, g_2), g_3) = +(g_1, +(g_2, g_3))$ holds.
- There exists an identity element in G. Specifically, there is an element $e \in G$ such that for all $g \in G$, $+(g, e) = +(e, g) = g$.
- Every element in G has an inverse. In other words, for each $g \in G$, there exists an element $(-g) \in G$ such that $+(g, -g) = +(-g, g) = e$.
- The operation "+" is commutative. This means that for any $g_1, g_2 \in G, +(g_1, g_2) = +(g_2, g_1)$. This property is what distinguishes an abelian group from a regular group.

The structure $(K, +, \cdot)$ is a *field* if it meets the following criterias:

1. $(K, +)$ is an abelian group,
2. $(K, \cdot)$ is an abelian group, and
3. the operation "$\cdot$" is distributive over "+". This means that for all $k_1, k_2, k_3 \in K$, the equation $\cdot\big(k_1, +(k_2, k_3)\big) = +\big(\cdot\,(k_1, k_2), \cdot(k_1, k_3)\big)$ is satisfied.

A *ring* is defined as an ordered triple $(R, +, \cdot)$, where the operations "+" and "$\cdot$" adhere to a set of ring axioms. These axioms are similar to those of a field, but they impose fewer constraints on the ordered pair $(R, \cdot)$. Specifically, this pair is only required to form an associative structure, as opposed to an abelian group as for fields.

Bits. An element from the set $\mathbb{Z}_2 = \{0, 1\}$ is referred to as a *bit*. A *bit string* is a vector that comprises n bits. The set of all n-bit strings is denoted as $\{0, 1\}^n$, while the set of all strings of any finite length, including potentially empty strings, is denoted as $\{0, 1\}^*$.

Classification. Classification is a process where a dataset is sorted into categories, and the term *classifier* is commonly used to refer to the technique employed for this categorization. In the context of classification, the input is a feature vector $a = (a_1, \ldots, a_n) \in \mathbb{R}^n$ that is classified using an evaluation function $f_m : \mathbb{R}^n \to \{k_1, \ldots, k_c\}$ based on a specific model m. The output, $k_t = f_m(a), t \in \{1, \ldots, c\}$, is the class to which the feature vector a is assigned according to the model m.

Vectors and inner product. In the context of an n-dimensional vector space with real number coordinates, denoted as $\mathbb{R}^n$, a vector takes the form $x = (x_1, \ldots, x_n)$, where $x_1 \ldots x_n \in \mathbb{R}$. The inner product, or dot product, between two vectors in $\mathbb{R}^n$ is defined as follows:

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i \tag{2.1}$$

This equation represents the sum of the products of the corresponding entries of the two sequences of numbers.

Hyper-plane decision classifier. The hyper-plane decision classifier is a classification technique. In this approach, the model m is constructed based on c vectors in $\mathbb{R}^n$, denoted as $m = \{m_i\}_{i=1}^{c}$. The output is determined as follows [5]:

$$k_t = \max_{i \in 1,\ldots,c} \langle m_i, a \rangle \tag{2.2}$$

Here, $\langle \cdot, \cdot \rangle$ signifies the inner product within a hypothesis space H that is equipped with an inner product. This equation essentially identifies the class k_t for which the inner product of the feature vector a and the model vector m_i is maximized.

Probabilities

Probability theory plays an important role in cryptography, as it enhances the generation of secure encryption keys by infusing a higher degree of randomness. This is accomplished by injecting uncertainty, which complicates and prolongs the process for a potential attacker trying to decipher the correct encryption key. Probability theory also serves to estimate the probability of an encryption key being cracked or guessed, allowing organizations to continually evaluate and modify their security measures as needed. In essence, probability theory bolsters cryptography by introducing an increased level of randomness and unpredictability. For instance, in symmetric key cryptosystems, the security of the system hinges on the randomness of the key. Probability theory can be employed to analyze the distribution of keys and the likelihood of an attacker guessing the correct key. In the realm of public key cryptography, probability theory is used to assess the security of the mathematical problems that underpin the encryption, such as the task of factoring large integers. The information about probabilities within this section are described according with [4, 6, 7].

Experiment. An experiment can be viewed as a process that yields one of a specified set of results. Each of these results is distinct. The potential results are referred to as simple events. The comprehensive set composed of all possible results is commonly known as the *sample space*. Usually, the discrete sample spaces that have a finite number of possible outcomes are used in cryptography. In this section, we will denote the simple events of a sample space as S, labeled as $s_1, s_2, \ldots, s_n$.

Probability distribution. A probability distribution K over S is characterized by a sequence of numbers $k_1, k_2, \ldots, k_n \geq 0$, where the sum of these numbers equals 1 ($k_1 + k_2 + \cdots + k_n = 1$). Each number o_i can be seen as the probability of the corresponding simple event g_i. This represents the outcome of the conducted experiment.

Event. An event E is a subset of the sample space S. In this context, the probability of occurrence of event E, denoted as $Pr[E]$, is defined as the sum of the probabilities o_i for all the simple events g_i that are part of E. If $g_i \in S$, $Pr[s_i]$ is simply denoted as $Pr[s_i]$.

Conditional probability. Assume E_1 and E_2 are two events, with $Pr[E_2] > 0$. The conditional probability of E_1 given E_2, denoted as $P(E_1|E_2)$, is expressed as follows:

$$Pr[E_1|E_2] = \frac{Pr[E_1 \cap E_2]}{Pr[E_2]}$$

$Pr[E_1|E_2]$ represents the probability of event E_1 occurring, given that event E_2 has already occurred.

Independent events: If E_1 and E_2 are two events, they are said to be independent if $Pr[E_1 \cup E_2] = Pr[E_1]Pr[E_2]$.

Bayes' Theorem. Given two events E_1 and E_2 with $P(E_2) \geq 0$, then

$$Pr[E_1|E_2] = \frac{(Pr[E_1]Pr[E_2|E_1])}{Pr[E_2]}$$

Random variable. Let X be a random variable, which is a function mapping from the sample space S to the set of real numbers. For each event $s_i \in S$, X assigns a real number, denoted as $X(s_i)$.

Probabilities. Given a finite set S, the notation $s \xleftarrow{\$} S$ denotes that s is selected randomly and uniformly from S. The symbol $Pr[E]$ signifies the probability of occurrence of event E. Similarly, $Pr[X = x]$ represents the probability that the random variable X assumes the specific value x. For a more comprehensive understanding of probabilities, one can refer to [8].

Lattices

The information about lattices within this section is according to definitions and explanations from [9].

A lattice is a mathematical structure that arises from the abstract study of order and structure. It is a set of elements that is closed under the operations of meet and join (which can be thought of as analogous to the operations of multiplication and addition in other algebraic structures).

Formally, a lattice $(\mathcal{L}, \leq)$ is a partially ordered set (poset) in which any two elements have a unique *supremum* (also called a least upper bound or join) and an *infimum* (also called a greatest lower bound or meet). An element a is said to cover an element b if $a > b$ and there is no element c such that $a > c > b$.

The meet ($\wedge$) and join ($\vee$) operations are defined as follows:

For any two elements $a, b \in L$, the meet of a and b, denoted by $a \wedge b$ is the greatest element in $\mathcal{L}$ that is less than or equal to both a and b.

$$a \wedge b = \max\{c \in \mathcal{L} | c \leq a \, and \, c \leq b\}$$

Similarly, the join of a and b, denoted by $a \vee b$, is the least element in $\mathcal{L}$ that is greater than or equal to both a and b.

$$a \vee b = \min\{c \in \mathcal{L} | a \leq c \, and \, b \leq c\}$$

In the context of cryptography, a lattice is a discrete, additive subgroup of Euclidean space. Lattices are used in various areas of cryptography, including the construction of cryptosystems, cryptographic protocols, and cryptographic functions.

Formally, a lattice $\mathcal{L}$ in the Euclidean space $\mathbb{R}^n$ is defined as the set of all integer linear combinations of n linearly independent vectors $b_1, b_2, ..., b_k \in \mathbb{R}^n$. These vectors form a basis $B = b_1, b_2, ..., b_k$ for the lattice.

Mathematically, a lattice L can be defined as:

$$\mathcal{L} = \mathcal{L}(B) = \Big\{ \sum_{i=1}^{k} \gamma_i b_i \mid \gamma_i \in \mathbb{Z}, b_i \in B \Big\}$$

where B is an $k \times k$ matrix whose columns are the basis vectors $b_1, b_2, \ldots, b_n$.

A parallelepiped associated with a basis of a lattice is the geometric object in the Euclidean space defined by the basis vectors. If we consider a lattice $\mathcal{L}$ in the Euclidean space $\mathbb{R}^n$ generated by the basis vectors $b_1, b_2, ..., b_k$, then the parallelepiped $\mathcal{P}(B)$ associated with this basis is the set of all points that can be expressed as a linear combination of the basis vectors, where the coefficients are real numbers in the interval $[-1/2, 1/2)$:

$$\mathcal{P}(B) = \Big\{ \sum_{i=1}^{k} x_i b_i \mid x_i \in [-1/2, 1/2) \Big\}$$

The *rank* of a lattice refers to the number of linearly independent vectors that generate the lattice, also known as the dimension of the lattice. In other words, it is the minimum number of basis vectors needed to span the lattice. Having the base B from above, the rank of the lattice $\mathcal{L} \subset \mathbb{R}^n$ is k. The rank of the lattice cannot exceed n, and a lattice is said to be *full-rank* if its rank is equal to n.

The volume (or determinant) of a lattice is a fundamental concept in lattice theory and is closely related to the determinant of the basis of the lattice.

Given a lattice $\mathcal{L} \subset \mathbb{R}^n$ generated by the basis vectors $B = b_1, b_2, ..., b_k$, the volume of the lattice, denoted as $Vol(\mathcal{L})$, is defined as the absolute value of the determinant of the matrix whose rows (or columns) are the coordinates of the basis vectors. In mathematical terms:

$$Vol(\mathcal{L}) = \sqrt{det(B^t B)}$$

The volume of a lattice provides a measure of the "density" of the lattice points in the space. It is also related to the size of the fundamental parallelepiped of the lattice, which is the region of space enclosed by the basis vectors and their negations.

In the context of lattice-based cryptography, the volume of the lattice is an important parameter that affects both the security and efficiency of the cryptographic scheme.

The *distance* in a lattice is a natural concept, and it is defined as the norm between two vectors: $dist(t, v) = || t - v ||, t\, in\, \mathbb{R}^n, v \in \mathcal{L}$. The minimum distance between an element $in\, \mathbb{R}^n$ and any element $v \in \mathcal{L}$ is defined as

$$dist(t, \mathcal{L}) = \min \{ || t - v || \mid v \in \mathcal{L} \}.$$

The minimum distance concept can be expanded as follows: $\lambda_1(\mathcal{L})$ is the minimum distance that involves the shortest vector:

$$\lambda_1(\mathcal{L}) = \min \{ || v || \mid v \in \mathcal{L}, v \neq 0.$$

This concept of distance can be generalized to $\lambda_i(\mathcal{L})$ which is the minimum distance to a non-zero vector of length i.

The set $span(\mathcal{L})$ refers to the vector space spanned by the lattice $\mathcal{L}$. The span of a lattice is the set of all linear combinations of the vectors in the lattice. If $\mathcal{L}$ is a lattice, then the span of $\mathcal{L}$ is given by:

$$span(\mathcal{L}) = \{a_1 b_1 + a_2 b_2 + \cdots + a_k * b_k \mid a_i \in \mathbb{R}\}$$

This set includes all vectors that can be reached by scaling and adding the basis vectors together.

The **dual** of a lattice, is another lattice consisting of all vectors in the vector space that have integer inner products with all vectors of the original lattice:

$$\mathcal{L}^* = \{v \in span(\mathcal{L}) \mid \langle v, b \rangle \in \mathbb{Z}, \forall b \in \mathcal{L}\}$$

where $\langle v, b \rangle$ denotes the inner product of vectors v and b. The concept of a dual lattice is important in the study of lattice-based cryptography, as many cryptographic problems can be related to the properties of the dual lattice.

An **ideal lattice** is a special type of lattice that arises from the ring of integers of a number field. More specifically, an ideal lattice is a lattice that is also a two-sided ideal of a ring. In the context of lattice-based cryptography, ideal lattices are particularly important because they allow for more efficient cryptographic algorithms. This is due to the additional algebraic structure that ideal lattices possess, which can be leveraged to design faster algorithms for problems such as the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP). An ideal lattice is denoted $\mathcal{L}(I)$ is an integer lattice $\mathcal{L}(B) \subseteq \mathbb{Z}^n$, with $B = \{g \mod f \mid g \in I\}$, $I \subseteq \mathbb{Z}[x]/\langle f \rangle$ is an ideal, and f is n-degree a monic polynomial.

The **Shortest Vector Problem (SVP)** is a well-known computational problem in lattice theory and is fundamental to many cryptographic systems, particularly those based on lattice cryptography. In its simplest form, the SVP asks for the shortest non-zero vector in a given lattice. SVP has several variants [10]:

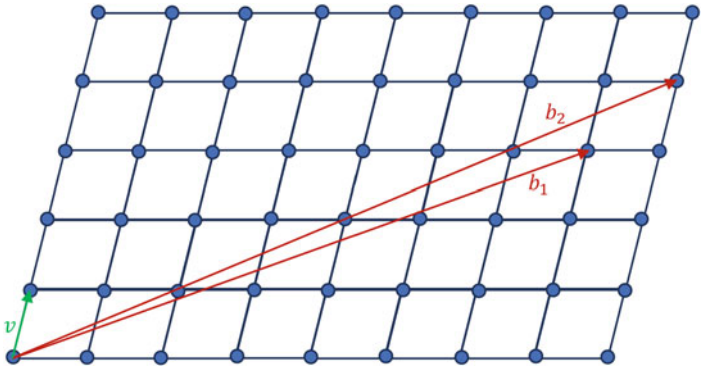

Fig. 2.1 Illustration of the SVP (inspired from research literature)

- γ-approximate SVP (SVP_γ): it requires to find the "almost" shortest vector: for $\gamma \geq 1$, find the vector $v \in \mathcal{L}$ that satisfies the condition $|| \ v \ || \leq \gamma \cdot \lambda_1(\mathcal{L})$. Figure 2.1 illustrates this case.
- γ-unique SVP (uSVP_γ): it requires finding the shortest vector that it γ times smaller than $\lambda_2(\mathcal{L})$: for $\gamma \geq 1$, find the vector $v \in \mathcal{L}$ for which $\gamma\lambda_1(\mathcal{L}) < \lambda_2(\mathcal{L})$
- Decisional SVP (DSVP or GapSVP_γ): it requires to decide which given bound for the shortest vector is the right one: for $\gamma \geq 1, r > 0$, decide whether $\lambda_1(\mathcal{L}) \leq r$ or $\lambda_1(\mathcal{L}) \geq \gamma \cdot r$.

The **Closest Vector Problem** (CVP) is another fundamental computational problem in lattice theory, which is also widely used in lattice-based cryptography. In the CVP, given a lattice $\mathcal{L}$ generated by a basis B and a target vector t not necessarily in the lattice, the problem is to find a vector v in $\mathcal{L}$ that is closest to t according to the norm. It has several variants [10]:

- γ-approximate CVP (CVP_γ): it requires to find the "almost" closest vector in a lattice to a target vector: for $\gamma \geq 1, t \in \mathbb{R}^n$ find the vector $v \in \mathcal{L}$ that satisfies the condition $dist(t, v) \leq \gamma \cdot dist(t, \mathcal{L})$. Figure 2.2 illustrates this case.
- α-Bounded Distance Encoding (BDD_α): it requires finding the closest vector $v \in \mathcal{L}$ to the target vector $t \in \mathbb{R}^n$ that satisfies the condition $dist(t, \mathcal{L}) < \alpha\lambda_1(\mathcal{L})$.
- Decisional CVP ($\text{DCVP}_{\gamma,r}$): it requires to decide, for $\gamma \geq 1, r > 0$, whether $dist(t, \mathcal{L}) \leq r$ or $dist(t, \mathcal{L}) \geq \gamma \cdot r$.

The **Learning With Errors (LWE)** problem is a problem in lattice-based cryptography that has been used as the foundation for many cryptographic systems. It was introduced by Oded Regev in 2005 [11] (a more comprehensive work [12]), which is an extension of [13]. The LWE problem is believed to be hard to solve, even for quantum computers. This makes it a good candidate for post-quantum cryptography, which aims to develop cryptographic systems that remain secure even in the presence of a quantum computer. The hardness of the LWE problem is based on the presumed difficulty of certain problems in lattice-based cryptography, such as the SVP and

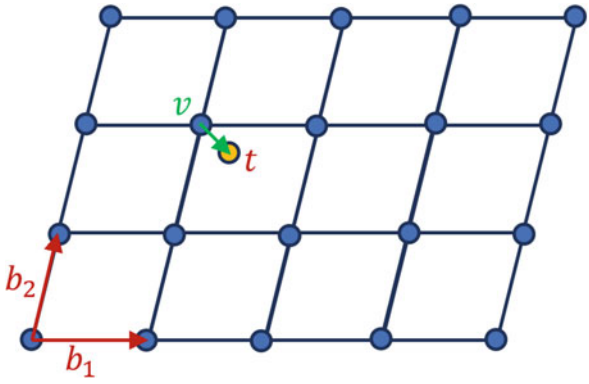

Fig. 2.2 Illustration of the CVP (inspired from research literature)

the CVP. This makes the LWE problem a popular choice for constructing secure cryptographic systems, as it provides a level of security that is based on well-studied problems in lattice-based cryptography.

LWE has two variants:

- Regular version (LWE): let $b \in \mathbb{Z}_q^m$ be a vector and $A \in \mathbb{Z}^{m \times n}$ be a matrix. In this setting, the LWE problem requires finding an unknown vector $s \in \mathbb{Z}_q^n$ that satisfies the condition $As + e = b \mod q$, where is randomly generated from $\mathbb{Z}_q^m$ related to a probability distribution χ. More specifically, find $s \in \mathbb{Z}_q^n$ based on a list of $m = n + 1$ noisy equations such that

$$A_{s,\chi} = \left\{(a_i, b_i = \langle a_i, s\rangle + e_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q \mid a_i \xleftarrow{\$} \mathbb{Z}_q^n, e_i \xleftarrow{\$} \chi\right\}.$$

- Decisional version (DLWE): it requires deciding whether m samples were chosen according to $A_{s,\chi}$ or according to $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

The **Ring-Learning With Errors (RLWE)** problem is the LWE version applied to rings, more specific it is developed under $(R_q)^2$, $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$, with q prime and $f(x) \in \mathbb{Z}[x]$ a d-degree monic, irreducible polynomial.

- RLWE: discover $s \in R_q$ based on arbitrary independent samples of $(a, b = s \cdot a + e) \in R_q \times R_q$, with $a \in R_q$ randomly uniformly chosen and $e \in R_q$ sampled from χ.
- Decisional RLWE (DRLWE): it requires to differentiate between independent and uniformly random samples from $R_q \times R_q$ and the same number of independent RLWE samples.

Another problem commonly used in lattice-based cryptography is the **Short Integer Solution Problem** (SIS) [14]. It is known for its hardness even against quantum computers, which makes it a popular choice for post-quantum cryptography. In the SIS problem, given a matrix A in $Z_q^{n \times m}$ (where Z_q is the ring of integers modulo q, n is the number of rows, and m is the number of columns), the goal is to find

a non-zero vector $x \in Z^m$ such that $Ax \equiv 0(\mod q)$ and $|| x ||$ is less than some bound β. The SIS problem is considered hard because, while it is easy to find a solution when β is large, it is computationally difficult to find a solution when β is small. This hardness is used to secure various cryptographic protocols. While in this form it does not seem, to be related to lattices, it can be transformed into a problem of finding a vector v that has the norm $|| v ||\leq \beta$ applied in the scaled dual lattice $\mathcal{L}_q^*(A)$ of $\mathcal{L}_q(A)$.

2.2 Cryptography Notations

This section introduces some of the key ideas in cryptography. This section's definitions and descriptions are based on [15–19] and these are listed in alphabetical order.

Adversary. The adversary, or attacker, is a probabilistic polynomial-time Turing computer that attempts to break the cryptosystem. In the analysis of cryptographic primitives, these are performed by a challenger that lets the adversary to train with the primitive in a specific way. For example, in symmetric encryption the challenger lets the adversary train with the encryption algorithm (without letting the adversary know the secret key). More specifically, the adversary chooses plain messages that it sends to the challenger and the challenger responds with the corresponding ciphertexts. This means that the adversary consults the encryption oracle. At some point, the adversary picks two different messages of the same length and the challenger gives the encryption for one of them. The purpose of the attacker is to decide for which of the two plain messages the ciphertext corresponds. However, in practice, the purpose of the adversary may differ from this approach. For example, the adversary can try to retrieve the secret key. Depending on the security model, the attacker may have other abilities, for example, to submit some types of queries to a challenger. An example of an attacker is a malicious entity that represents an internal entity trying to compromise the systems/information, by behaving as an authorized/trusted entity. A curious entity analyzes the information, but it does not intervene (see an example of an honest-but-curious server below).

Asymmetric encryption. *Public-Key Cryptography* (PKC) is another name for *asymmetric encryption*. There are two sorts of keys used in these schemes: the *public* key for encryption and the *secret* or *private* key for decryption.

Attack model. The attack model specifies what an opponent is capable of doing inside the system and how the system behaves when the attacker is there. The aim of the assault specifies what the opponent is attempting to accomplish while acting in accordance with the model. Examples of assault models include: chosen plaintext attack, chosen ciphertext attack, keyword guessing attack, etc.

Authorities and third parties. In most cases, the authorities are met while granting certificates. As a result, a certificate authority is an authorized body that issues digital certificates. Other entities seen in cryptography include *trusted* third parties—a trusted third party is an entity that acts as an intermediary between two entities

that both confide in the third party. Certification authorities, registration authorities, timestamp authorities, and so forth are examples.

Biometric features. These are physical characteristics that uniquely identify a person in cryptography. The biometric characteristics are typically employed in the system authentication process, but they may also be used in the encryption process. Examples of biometric features are the fingerprints, the iris, the face, the voice, etc.

Circuit. A circuit is a connected graph with nodes designed by Boolean operators. An example of a circuit is the Boolean circuit that is constructed using just two types of gates, namely the addition (Boolean "or") and the multiplication (Boolean "and") whose input is one bit $b \in \{0, 1\}$. A useful gate is $NAND$, because it can be used to construct any computation.

Challenger. It is a hypothetical probabilistic algorithm that interacts with an attacker in a game-based security paradigm (see the *game* explanation below). The challenger creates the encryption system's keys and responds to the attacker's requests (if any exist).

Cryptosystem. A cryptosystem represents a tuple of five elements $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, where $\mathcal{P}, \mathcal{C}, \mathcal{K}$ are finite sets that represent the plaintexts set, the ciphertexts set and the key space, respectively. The element $\mathcal{E}$ represents the set of encryption rules and the element $\mathcal{D}$ represents the set of the decryption rules, such that $e_K : \mathcal{P} \to \mathcal{C}, e_K \in \mathcal{E}$ and $d_K : \mathcal{C} \to \mathcal{P}, d_K \in \mathcal{D}$ have the following property:

$$d_K(e_K(x)) = x, \forall x \in \mathcal{P} \tag{2.3}$$

In practice, a cryptosystem has a slightly different form: $\epsilon =$ `(KeyGeneration, Encryption, Decryption)`. These are four sub-algorithms that represent the key generation algorithm that is a probabilistic polynomial-time algorithm (here, the output can be just one key, used for both encryption and decryption, therefore it results in symmetric encryption; or the output can be a pair of keys, namely the public key used for encryption and the private key used for decryption, therefore it results in asymmetric encryption), the encryption algorithm that is a probabilistic polynomial-time algorithm, the decryption algorithm that is a deterministic polynomial-time algorithm, respectively. According to the type of encryption (for example, searchable or homomorphic encryption), we will see in the following chapters that a cryptosystem may contain additional algorithms.

Game. The game demonstrates one method to create a security model. In a game, the challenger generates the encryption system's keys, and the attacker may send different requests to the challenger with certain limits, to which the challenger must reply. The game finishes when the attacker completes its activity; at this time, it is possible to determine if the attacker accomplished what was required to break the encryption scheme. Assume the attacker did not defeat the game's encryption scheme. In such instance, the likelihood of an arbitrary attacker breaking the encryption system must be calculated. Obviously, this probability must be extremely low in order to declare the encryption scheme safe.

Hardness assumption. It is a computational assumption that claims that a given mathematical problem cannot be addressed efficiently, i.e. in polynomial time,

respecting anything. Examples of hardness assumptions are: discrete logarithm problem, Diffie-Hellmann assumptions, cryptographic pairings, etc.

Honest-but-curious server. The servers with this attribute follows the protocols as they should, but it may also probe the data in order to get more information.

Leakage function. A leakage function displays what is leaked into dynamic searchable encryption systems' search, add, or remove algorithms.

Negligible function. Considering $k \in \mathbb{N}$, a function $g : \mathbb{N} \to \mathbb{N}$ is called negligible in k if for any $p(\cdot)$ and any k large enough, the condition $g(k) < 1/p(k)$ is fulfilled. In this definition, $p(\cdot)$ represents a positive polynomial. The notation $f(k) = negl(k)$ means that it exists a negligible function $g(\cdot)$ such that the condition $f(k) \leq g(k)$ is fulfilled for any k large enough.

Oblivious RAM (ORAM). An ORAM (Oblivious RAndom Access Memory) is a simulator that transforms an algorithm such that it preserves the behavior of the input, but at the same time it is "memory oblivious" (the access pattern to the memory is independent of the input).

Protocol. A cryptographic protocol between n parties represents a procedure (therefore, based on a set of rules) of messages exchanging between the n parties with the goal of satisfying a security property (confidentiality, integrity, authentication, non-repudiation, etc.). The messages are constructed over public sets and/or private elements using different cryptographic primitives. The communication procedure between the n parties may or may not require sequential communication.

Provable security. It represents a methodology used for analyzing the security of encryption systems. When for an encryption system the security requirements can be expressed formally for an adversarial model (for which it is clearly specified that the adversary can access the system and has sufficient computational resources) then it is called that the system has provable security. The proof of the security means to prove that the security requirements are met under the assumptions for the adversary's capabilities and some of the clear assumptions regarding the *hardness* (see above) of a computational task are held.

Pseudo-random number generator (PRNG). A PRNG represents a function that, when it receives an initial input value called *seed*, outputs a sequence of numbers, a sequence that seems to be random for an arbitrary observer. These are widely used in different cryptographic applications, for example in key generation algorithms or various components of protocols.

Security goal. A security goal is a target used to evaluate the security of the encryption system. Examples of security goals are indistinguishability, non-malleability, non-decidability, and plaintext awareness.

Security model. A security model is a pair that contains the security goal and the attack model.

Security parameter. Security analysis of an encryption system is made by estimating the advantage of a generic probabilistic polynomial-time algorithm against the system. This advantage represents the probability expressed in relation to a parameter of the input data. Such parameter is used by the key generation algorithm and it is called *security parameter*. Because a security parameter is used to estimate the complexity of a probabilistic polynomial-time Turing algorithm and because the

representation of the numbers for Turing machines is unary, often the notation 1^k is used to specify the security parameter, where $k \in \mathbb{N}$. To simplify the notation, the security parameter will be denoted simply k.

Symmetric encryption. This type of encryption scheme uses the same key for both encryption and decryption rules. This key is called *private key* or *secret key*.

2.3 Technologies

2.3.1 Cloud Computing and Big Data

Cloud computing is a model for delivering computing services where resources are retrieved from the Internet through web-based tools and applications, rather than directly from physical devices, such as servers. It allows individuals and businesses to access and use the resources on-demand, without the need for physical infrastructure or direct management. It has become increasingly popular in various industries, enabling organizations to focus on their core business activities, while relying on cloud service providers to handle the underlying infrastructure and technical aspects.

Nowadays, cloud computing represents an important component of the technology, providing all services as on-demand utilities. The main characteristics of the cloud are the following [20]:

- *Virtualization*. The central infrastructure may provide the systems and storage, and the resources can be shared and distributed. These characteristics allow the resources to be scaled, the expenses to be evaluated on a measurable basis, and numerous tenants to exist and utilize the resources at the same time.
- *Abstraction*. Cloud services are implemented in an abstracted form, relieving users and developers from the implementation process. This abstraction grants users access to seemingly limitless resources, while data is stored in unspecified locations. The services operate on undisclosed physical machines, and the management of systems is entrusted to external organizations, known as cloud providers, who assume responsibility on behalf of the data owner/user.

In cloud computing, a service refers to a specific offering or solution provided by a cloud service provider. These services are designed to meet different computing needs, such as storage, processing power, software applications, and networking capabilities. The main categories of services are:

- *Software as a Service* (*SaaS*): In this service model, cloud providers deliver software and applications through the Internet. Users subscribe to the software and access it via the web or vendor APIs.
- *Platform as a Service* (*PaaS*): This involves providing a platform allowing customers to develop, run, and manage applications without needing to build and maintain the infrastructure typically associated with application development and launch.

- *Infrastructure as a Service* (*IaaS*): This involves providing (virtualized or otherwise) hardware, storage, servers, and data center space or network components to users.

Classical examples of well-known cloud computing services are Google's G Suite and Google Cloud Platform, Amazon Web Services (AWS), Microsoft Azure, and IBM Cloud.

However, there are many other several types of services, but these can be broadly included in the main services from above. Other examples of cloud services are: *Function as a Service* (FaaS—examples: AWS Lambda, Azure Functions, and Google Cloud Functions), *Database as a Service* (DBaaS—examples: Amazon RDS, Azure SQL Database, and Google Cloud Spanner), *Backend as a Service* (BaaS, examples: Firebase by Google and AWS Amplify), *Containers as a Service* (CaaS—examples: Amazon Elastic Kubernetes Service (EKS), Azure Kubernetes Service (AKS), and Google Kubernetes Engine (GKE)), *Desktop as a Service* (DaaS—examples: Amazon WorkSpaces, Windows Virtual Desktop (WVD), and VMware Horizon Cloud), *Security as a Service* (SECaaS—examples: cloud-based firewall services, cloud-based antivirus solutions, and cloud-based security information and event management (SIEM) systems).

Taking into consideration its virtualization and abstraction, cloud computing offers several advantages:

- *Scalability*: Cloud resources can be scaled up or down based on demand, allowing businesses to quickly adapt to changing needs without significant investments or delays.
- *Flexibility*: Users can access cloud services and applications from anywhere with an Internet connection, using a variety of devices, including computers, smartphones, and tablets.
- *Cost Savings*: Cloud computing eliminates the need for organizations to invest in and maintain their own hardware and infrastructure, reducing expenses. It also offers a *pay-as-you-go* model, where users pay only for the resources they consume.
- *Reliability and Availability*: Cloud service providers typically have robust infrastructure with strong systems and data policies, ensuring high availability and minimizing the risk of data loss or service disruptions.
- *Collaboration and Integration*: Cloud computing enables teams to collaborate on projects in real-time, as data and applications can be easily accessed and shared. It also allows integration with other cloud-based or on-premises systems, enabling organizations to leverage existing tools and services.

A technology that stands at the top of cloud computing is *big data*. Big data is referring to large and complex sets of data that are difficult to process and analyze using traditional data processing methods, such as traditional database management systems. Big data includes large amounts of structured, semi-structured, and unstructured data, which is generated from different sources such as social media, sensors, machines, and transactions. Big data is characterized by the *volume* (large amounts

of data), *velocity* (the fast speed at which data is generated and processed), and *variety* of data (different types of data from wide ranges of sources). These characteristics are known as the "3V", but often there are more Vs: veracity, variability, value, visualization, etc. Having these features, it can be easily seen that big data often requires specialized tools and techniques to extract insights and value from the data. Therefore, the technology that can provide the infrastructure and resources necessary to store, process, and analyze large volumes of data, facts that are the essence of big data., is cloud computing. Big data is so important because it provides tools for other emerging technologies that raised in the last few years, such as AI, augmented reality (AR), virtual reality (VR), blockchain, drones, IoT, robotics, 3D printing, Internet of Behavior (IoB), hyperautomation, edge artificial intelligence, 5G, cybersecurity mesh, or distributed cloud. These technologies have the potential to impact businesses and industries through advanced data analysis, automation, enhanced connectivity, and even improved security.

Having this context, it can be easily seen that security is a significant challenge for cloud computing and big data due to the large amount of data involved and the complexity of the systems. Data breaches represent a high risk, as they can lead to unauthorized access and exposure of sensitive data, resulting in severe consequences for individuals and organizations. Data loss or leakage is another critical security issue, as the massive volumes of data stored and processed in these environments increase the likelihood of accidental or intentional data exposure. Lack of encryption is a common vulnerability, leaving data susceptible to interception and unauthorized decryption. Insecure APIs further exacerbate the security challenges, as they can be exploited by attackers to gain unauthorized access to the cloud infrastructure or manipulate data. Insider threats also pose a substantial challenge, as employees or insiders with malicious intent can exploit their access privileges to compromise data security. Additionally, compliance and regulatory issues add complexity, as organizations must navigate various legal and industry-specific requirements to ensure data privacy and protection.

Cloud computing, due to its complex nature, inherits the security challenges of its components. These security issues can be categorized based on where they occur, as outlined by Ali et al. [21]. These categories include the communication level, the architectural level, and the contractual level, which encompasses legal aspects. Services in cloud computing are accessed via the Internet, which involves various protocols and protective mechanisms for the communication channel between the user and the service source in the cloud. This also applies to communication between cloud components. At this stage, any security issues that can occur in external or internal communication are inherited by the cloud. In terms of architecture, elements such as web applications, Application Programming Interfaces (APIs), virtual machines, and data components are included. These components also pass on their security issues to the cloud. The contractual level involves concepts like Service Legal Agreements (SLAs), but it does not encompass security issues related to technical aspects. In the following sections, we will present common examples of security issues in cloud computing:

- *Data Breaching*: This security issue involves unauthorized entities (e.g., malicious users, applications, or services) illegitimately viewing, accessing, or retrieving data. Data breaching results in data theft or its retrieval to an unsecured or illegitimate storage location.
- *Exploiting System Vulnerabilities*: While not a new type of security issue, system vulnerability exploitation has become a dangerous reality as entities (e.g., individuals, companies, institutions) share or exchange data, thereby creating new points of vulnerability. Continuous or periodic system scanning or other similar basic methods can help avoid system exploitation.
- *Shared Resources*: At the top of the cloud services stack are the applications, followed by the platforms, and finally the infrastructures. All these types of services may use the same memory. Therefore, when a vulnerability is identified in one of these layers, it can affect all other layers if the shared memory or resources are compromised. This behavior can lead to a chain of vulnerabilities if not properly addressed [22].
- *Virtualization*: Virtualization introduces a broad spectrum of vulnerabilities [23], thereby creating new potential attack vectors. The weak points of virtualization include the hypervisor, the public repository of images for virtual machines, operations on virtual machines (e.g., rollback or life cycle), and the virtual networks. In cloud computing, the hypervisor is the primary virtual machine that manages the use of shared resources by multiple virtual machines. If the hypervisor is compromised, all virtual machines under its control can also be compromised. The public repository of images for virtual machines contains the configuration settings files for the system's virtual machine, playing a crucial role in cloud computing security. Concerning operations on virtual machines, the rollback operation can revert a virtual machine to a previous state with a known security issue that was resolved in a later state. This means that reverting a virtual machine to a previous state could reintroduce a previously resolved security issue. Furthermore, the non-deterministic behavior of virtual machines can significantly impact the proper functioning of the virtual machine network. For instance, the hypervisor can suspend, activate, or deactivate a virtual machine, or the virtual machine could malfunction. These operations can affect the rest of the network's virtual machines or communication, revealing new potential attack vectors.

As the use of cloud computing expands, big data also evolves, amassing a vast amount of data from a wide variety of sources. In fact, any device that generates any form of data contributes to big data collection.

2.3.2 *Artificial Intelligence and Machine Learning*

In the previous section, we have seen that big data involves processing large amounts of information. In the last few years, machine learning and artificial intelligence have gained attention in this regard, becoming nowadays basically part of our lives.

We are surrounded by AI tools, starting with smart assistants on mobile devices, such as Google Assistant, Alexa from Amazon, or Siri from Apple, and ending with AI tools (popular at the moment of writing this books) for text generating like ChatGPT,[1] Jasper,[2] video generators like Pictory,[3] Synthesys,[4] image generators like Images.ai,[5] voice generator and text-to-speech platform like Lovo.io,[6] Speechify[7] and the examples can go on.

Machine learning and artificial intelligence have significantly transformed the way we interact with technology and data. ML involves the development of algorithms that allow computers to learn from and make decisions or predictions based on data. AI, on the other hand, is a broader concept that involves creating machines or software that can perform tasks that would normally require human intelligence, such as understanding natural language, recognizing patterns, and making decisions.

The impact of ML and AI on society and various industries has been significant. They have revolutionized sectors such as healthcare, used to predict disease outcomes and personalize treatment plans; finance, used for fraud detection and credit scoring; and transportation, used in autonomous vehicles and traffic management systems. They have also significantly enhanced our ability to interact with technology through voice assistants, recommendation systems, and personalized content.

Taking into consideration these aspects, the relationship between ML/AI, big data, and cloud computing is symbiotic. Big data provides the vast amounts of information that ML algorithms need to learn and make accurate predictions. Cloud computing, with its virtually unlimited storage and computational power, provides the infrastructure necessary to store this data and run complex ML algorithms. Without big data and cloud computing, the current advancements in ML and AI would not be possible.

However, integrating ML/AI with big data and cloud computing presents several security issues and challenges. One of the main concerns is data privacy. As more data is collected and analyzed, there is an increased risk of sensitive information being exposed or misused. Moreover, ML models can be vulnerable to adversarial attacks, where attackers manipulate the input data to deceive the model and cause it to make incorrect predictions. Another challenge is the lack of transparency and interpretability in many ML models, often referred to as the "black box" problem. This can make it difficult to understand why a model is making certain predictions, which can be problematic in fields where explainability is an informant factor, such as healthcare and finance. The reliance on cloud computing for storing data and running ML models introduces additional security risks. Data stored in the cloud can be vulnerable to breaches, and cloud-based ML models can be targeted by attacks aiming

[1] https://chat.openai.com.

[2] https://www.jasper.ai.

[3] https://pictory.ai.

[4] https://synthesys.io.

[5] https://images.ai.

[6] https://lovo.ai.

[7] https://speechify.com.

to disrupt their operation. To address these challenges, researchers are developing new techniques and technologies, such as differential privacy to protect sensitive data, adversarial training to make ML models more robust to attacks, and explainable AI to increase the transparency of ML models. Moreover, secure cloud architectures and encryption techniques are being used to protect data stored in the cloud. More about machine learning applied in cybersecurity can be found in [24].

2.3.3 Internet of Things

The Internet of Things (IoT) refers to a network of interconnected physical objects, devices, machines, or living beings that are equipped with sensors, software, and other technologies which collect and exchange data over the Internet. These "things" can communicate with each other and with computing systems without the need for human interaction. IoT technology enables the gathering and analysis of vast amounts of data, which can be used to improve efficiency, automate processes, enhance decision-making, and enable new services and applications across different industries. By connecting devices and objects to the Internet, IoT brings several advantages: real-time monitoring and control, predictive maintenance, improved asset management, energy efficiency, enhanced customer experiences, etc.

By far, cloud computing, big data, machine learning, and artificial intelligence are relevant technologies for IoT. However, the following can be added on the list:

- Edge Computing, which makes data processing and analysis be performed closer to the edge devices, reducing latency and improving efficiency in IoT systems.
- Blockchain, which provides secure and decentralized data storage and transaction validation, enhancing data integrity and trustworthiness in IoT applications (see Sect. 3.6.3 for more details).
- Virtual Reality and Augmented Reality, which can enhance user experiences by integrating virtual and real-world elements, enabling applications such as remote monitoring and maintenance in IoT systems.
- Robotics: IoT can be combined with robotics to create intelligent and autonomous systems that can perform tasks in different industries, such as manufacturing, healthcare, and agriculture.
- 5G: the fifth generation of wireless technology is relevant to IoT through improved connectivity and faster data transfer.

Even if IoT is an important technology nowadays, it also presents challenges related to data privacy, security, compatibility, and scalability. More about security challenges in IoT systems can be found in [25].

2.3.4 Blockchain

Blockchain technology has gained importance in recent years due to its potential to revolutionize different industries. At its core, a blockchain is a decentralized and distributed digital ledger that records transactions across multiple devices. It operates on a peer-to-peer network, where each participant, known as a node, has a copy of the entire blockchain. This eliminates the need for a central authority, therefore trust and transparency among participants are enabled.

The key concept behind blockchain is the creation of a block, which contains a batch of transactions. These blocks are linked together using cryptographic hash functions, forming a chain of blocks, hence the name of the blockchain. Once a block is added to the chain, it is nearly impossible to alter or delete the information, making blockchain tamper-resistant and immutable, because each participant has o copy of it.

One of the primary advantages of blockchain is its ability to provide transparency. Every transaction recorded on the blockchain is visible to all participants, creating an auditable and accountable system. This property can be particularly beneficial in industries such as supply chain management, where tracking the movement of goods becomes more efficient and trustworthy. Another aspect of blockchain is its enhanced security derived due to the use of cryptographic algorithms that ensure the data stored on the blockchain remains secure and tamper-proof. Additionally, the decentralized nature of blockchain makes it resistant to single points of failure and cyber attacks. This feature is especially relevant in financial transactions, where the need for secure and trustworthy systems is crucial. Blockchain technology has practical applications in different industries, including finance, healthcare, and logistics. In the financial sector, blockchain can streamline and automate processes, reducing costs and eliminating intermediaries. Smart contracts (self-executing agreements embedded within the blockchain) are the basis for secure and efficient transactions without the need for traditional legal frameworks.

However, despite its promises, blockchain faces several challenges and issues. One significant concern is scalability. As more transactions are added to the blockchain, the network can become congested, leading to slower transaction times and increased costs. Efforts are being made to develop solutions like off-chain transactions. Another challenge is regulatory and legal compliance because as blockchain operates on a global scale, different jurisdictions may have conflicting regulations. Interoperability is also a challenge that blockchain needs to overcome. With multiple blockchain platforms and protocols, ensuring seamless communication and compatibility between different chains is essential for its large-scale adoption.

While this transparency is one of the strengths of blockchain, it can also raise privacy concerns, especially when sensitive data is stored on the blockchain. Homomorphic encryption can address these concerns by allowing data to remain encrypted while being processed or verified. Using holomorphic encryption, transactions can be verified without revealing the actual transaction data. For example, it can be used to prove that a transaction is valid (i.e., the sender has enough funds) without reveal-

ing the exact value of the transaction or the total balance of the sender. This can be particularly useful in scenarios where privacy is a concern, such as in confidential transactions in financial services. More information about blockchain can be found in [26].

References

1. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (2018)
2. Meijer, A.R.: Algebra for Cryptologists. Springer International Publishing (2016)
3. Gilbert, L.: Elements of Modern Algebra. Cengage Learning (2014)
4. Simonnet, M.: Measures and Probabilities. Springer Science & Business Media (2012)
5. Nasrabadi, N.M.: Pattern recognition and machine learning. J. Electron. Imaging **16**(4), 049901 (2007)
6. Silverman, J.H., Pipher, J., Hoffstein, J.: An Introduction to Mathematical Cryptography. Springer (2008)
7. Tabak, J.: Probability and Statistics: The Science of Uncertainty. Infobase Publishing (2014)
8. Evans, M.J., Rosenthal, J.S.: Probability and statistics: The science of uncertainty. Macmillan (2004)
9. Peikert, C.: A decade of lattice cryptography. TCS **10**, 283–424 (2016)
10. Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: Matsui, M. (ed.) Advances in Cryptology—ASIACRYPT 2009, pp. 617–635. Springer (2009)
11. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**, 34:1–34:40 (2009)
12. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-quantum Cryptography, pp. 147–191. Springer (2009)
13. Blum, A., Furst, M., Kearns, M., Lipton, R.J.: Cryptographic primitives based on hard learning problems. In: Stinson, D.R. (ed.) Advances in Cryptology—CRYPTO' 93, pp. 278–291. Springer (1994)
14. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, pp. 99–108. Association for Computing Machinery (1996). https://doi.org/10.1145/237814.237838
15. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. J. ACM (JACM) **43**, 431–473 (1996)
16. Dent, A.W.: Fundamental problems in provable security and cryptography. Philos. Trans. R. Soc. A: Math. Phys. Eng. Sci. **364**, 3215–3230 (2006)
17. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 965–976 (2012)
18. Larsen, K.G., Nielsen, J.B.: Yes, there is an oblivious RAM lower bound! In: Advances in Cryptology-CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II, pp. 523–542. Springer (2018)
19. Stinson, D.R., Paterson, M.: Cryptography: Theory and Practice. CRC Press (2018)
20. Velte, A., Velte, T., Elsenpeter, R.: Cloud Computing. McGraw-Hill (2010)
21. Ali, M., Khan, S.U., Vasilakos, A.V.: Security in cloud computing: opportunities and challenges. Inf. Sci. **305**, 357–383 (2015)
22. Ryan, M.D.: Cloud computing security: the scientific challenge, and a survey of solutions. J. Syst. Softw. **86**, 2263–2268 (2013)
23. Hashizume, K., Rosado, D.G., Fernández-Medina, E., Fernandez, E.B.: An analysis of security issues for cloud computing. J. Internet Serv. Appl. **4**, 1–13 (2013)

24. Martínez Torres, J., Iglesias Comesaña, C., García-Nieto, P.J.: Machine learning techniques applied to cybersecurity. Int. J. Mach. Learn. Cybern. **10**, 2823–2836 (2019)
25. Mishra, N., Pandya, S.: Internet of things applications, security challenges, attacks, intrusion detection, and future visions: a systematic review. IEEE Access **9**, 59353–59377 (2021)
26. Gayvoronskaya, T., Meinel, C.: Blockchain. Springer (2021)

Chapter 3
Homomorphic Encryption

3.1 History

In homomorphic encryption, computations are performed directly over the encrypted data, reducing the need for decryption during computations. The most essential aspect is that the result of the decrypted cyphertext obtained after a specific computation on encrypted data must be identical to the result obtained by applying over ordinary data some computations that reflect the computations applied on encrypted data. Third parties are able to evaluate functions on encrypted data without having access to unencrypted data, which is a significant capability enabled by HE schemes. Applying functions results in applying algorithms, allowing for the execution of various analysis algorithms, such as machine learning, data science, statistics, artificial intelligence, etc. HE schemes simplify the flow of actions required to perform computations on data, while preserving privacy and security. A practical and illustrative example is updating a cloud-based encrypted distributed database. Beginning with the financial/business and healthcare sectors and concluding with the government field, homomorphic encryption has a wide range of applications. The concept of homomorphic encryption (HE) was introduced in the late 1970s with the advent of modern cryptography, but its full realization took several decades. See Fig. 3.1 for the timeline of HE and the description above for more details.

Late 1970s—Early Foundations. The idea was first proposed by Rivest, Adleman, and Dertouzos in 1978, not long after the invention of Rivest-Shamir-Adleman (RSA), one of the first practical public-key cryptosystems. They introduced the concept of a *privacy homomorphism* [1], which would allow specific computations on encrypted data without needing to decrypt it first.

1990s—Partial Homomorphic Encryption. In 1996, the Diffie-Hellman key exchange [2], a method of securely exchanging cryptographic keys over a public channel, was found to have a multiplicative homomorphic property. Other cryptographic systems such as RSA [3] and ElGamal [4] were also discovered to be either multiplicatively or additively homomorphic, but not both. These systems were

S. L. Nita and M. I. Mihailescu, *Advances to Homomorphic and Searchable Encryption*,
https://doi.org/10.1007/978-3-031-43214-9_3

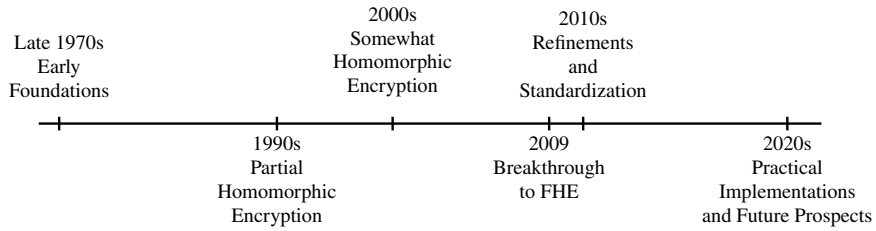

Fig. 3.1 Timeline of the history of homomorphic encryption

defined as *Partial Homomorphic Encryption* (PHE) schemes because they supported operations for only one arithmetic operation, either addition or multiplication.

2000s—Somewhat Homomorphic Encryption. By the 2000s, advancements led to the emergence of *Somewhat Homomorphic Encryption* (SHE) schemes. These could handle a limited number of both addition and multiplication operations, but still fell short of providing an entirely practical solution due to their limitation on the depth of computation.

2009—Breakthrough to Fully Homomorphic Encryption. The turning point came in 2009, when Craig Gentry, a computer scientist at IBM, presented the first Fully Homomorphic Encryption scheme as part of his Ph.D. thesis [5]. FHE allows arbitrary computations on encrypted data, providing a solution to the limitations of PHE. However, the initial design was quite inefficient and not practical for real-world applications due to its high computational overhead.

2010s—Optimizations and Standardization. In the following decade, extensive research was undertaken to improve the efficiency and practicality of FHE schemes. Gentry, along with other researchers, proposed several optimized versions of his initial construction to decrease its complexity and computational requirements. By 2020, advancements in HE made it more applicable for practical use-cases like secure data analysis, private information retrieval, and secure voting systems. During this period, efforts were also initiated to standardize homomorphic encryption under organizations like HomomorphicEncryption.org.

2020s—Practical Implementations and Future Prospects. As of 2023, homomorphic encryption continues to gain traction as a tool for secure computations on sensitive data in various fields, from healthcare to finance. It remains a thriving area of cryptographic research and continues to evolve in terms of efficiency, security, and usability. Despite its computational challenges, the continuous improvements in hardware and software technologies, along with cryptographic optimizations, hint at a promising future for this revolutionary encryption technique.

3.2 Definitions

The purpose of homomorphic encryption is to facilitate direct computations on the encrypted data. Therefore, the confidentiality of the data is maintained during processing, even if it is outsourced or analyzed by third parties.

In homomorphic encryption, only the operations of *addition* and *multiplication* must be homomorphic because any function can be expressed as a circuit based on *or* and *and* gates (corresponding to addition and multiplication operations). It is important to notice that subtraction is the inverse operation of addition and division is the inverse operation of multiplication.

In this section, we present the principal definitions for the cryptography topic of homomorphic encryption, based on well-known contributions to this field [6–8].

Definition 3.1 A function $f : A \rightarrow B$ is referred to as ***homomorphism*** if it possesses the following characteristic:

$$f(x_1 \circ x_2) = f(x_1) * f(x_2), \forall x_1, x_2 \in A \tag{3.1}$$

where "$\circ$" is a binary operation on A, "$*$" is a binary operation on B and A, B are algebraic structures of the same signature (category). For example, A and B are both groups, rings, or fields.

Definition 3.2 A ***homomorphic encryption scheme*** ε consists of a tuple of four probabilistic polynomial-time algorithms $\varepsilon =$ `(KeyGeneration, Encryption, Decryption, Evaluate)`, where each of the has a well-established purpose:

- `KeyGeneration` $(\lambda, 1^{\tau}) \rightarrow (k_s, k_p)$: The security parameter λ and the functionality parameter τ (see more about τ in Definition 3.4) represent the input, while the output is a tuple of the secret key k_s and public key k_p.
- `Encryption` $(m, k_p) \rightarrow c$: The plain bit message $m \in \{0, 1\}$ and the public key k_p represent the input, while the output is the encrypted message (ciphertext) $c \in \{0, 1\}$.
- `Decryption` $(c, k_s) \rightarrow m$: The encrypted text $c \in \{0, 1\}$ and the secret key k_s represent the input, while the output is the plain message $m \in \{0, 1\}$.
- `Evaluate` $(k_p, \pi, \overrightarrow{v}) \rightarrow \overrightarrow{v}'$: The public key k_p, a circuit π and a vector of encrypted texts $\overrightarrow{v} = (v_1, \ldots, v_z)$ represent the input, where every v_i, $1 \leq i \leq z$ corresponds to each entrance bit of π, while the output is represented by a vector of encrypted texts $\overrightarrow{v}' = (v'_1, \ldots, v'_t)$, where every v'_j, $1 \leq j \leq t$ corresponds to each exit bit of π.

A note regarding Definition 3.2: depending on their specificity, the key generation algorithm may employ just one of the parameters. The encryption and decryption algorithms both accept a number of vectors of plain bits and vectors of encrypted bits, respectively, as arguments. The encrypted texts produced by the `Encryption` algorithm are referred to as *fresh ciphertexts*, whereas the encrypted texts produced

by the `Evaluation` algorithm are referred to as *evaluated ciphertexts*. The property of homomorphism may be observed here: The `Evaluation` algorithm employs the computations made on the ciphertext, while the circuit employs the computations made on the plaintext. This can be easily seen from the correctness property from Definition 3.3.

Definition 3.3 Let $\varepsilon =$ `(KeyGeneration, Encryption, Decryption, Evaluate)` be a homomorphic encryption scheme and $C = \{C_\tau\}_{\tau \in \mathbb{N}}$ a family of circuits. The homomorphic encryption scheme ε is called ***(perfectly) correct*** for C if $\forall \lambda, \tau \in \mathbb{N}$ if it satisfies the following conditions:

1. $\forall b \in \{0, 1\}$,

$$Pr\Big[\texttt{Decryption}(k_s, c) = b \mid \big(k_s, k_p\big) \leftarrow \texttt{KeyGeneration}(1^\lambda, 1^\tau), \\ c \leftarrow \texttt{Encryption}(k_p, b)\Big] = 1 \tag{3.2}$$

2. $\forall \pi \in C_\tau$ and $\forall \overrightarrow{b} = (b_1, \ldots, b_z) \in \{0, 1\}^z$, with b_i corresponding to the π_i entrance of π,

$$Pr\Big[\texttt{Decryption}(k_s, \overrightarrow{c}') = \overrightarrow{b} \mid \big(k_s, k_p\big) \leftarrow \texttt{KeyGeneration}(1^\lambda, 1^\tau), \\ \overrightarrow{c} \leftarrow \texttt{Encryption}(k_p, \overrightarrow{b}), \overrightarrow{c}' \rightarrow \texttt{Evaluate}(k_p, \pi, \overrightarrow{c})\Big] = 1 \tag{3.3}$$

The two conditions outlined in Definition 3.3 indicate that the homomorphic encryption ε provides accurate decryption for both fresh and evaluated ciphertexts. The broadest range of circuits, C, where the homomorphic encryption ε meets the condition stated in Definition 3.3, is referred to as the *homomorphic capacity of* ε. While these conditions can be slightly modified to admit a negligible probability, resulting in less stringent requirements, many constructions frequently use these stricter criteria.

Semantic security for homomorphic encryption schemes, as defined in Definition 3.4, is typically addressed in the context of key generation, encryption, and decryption algorithms, as explained in [9]. To deem the `Evaluation` algorithm as secure (which is a public algorithm), a probabilistic polynomial-time (PPT) adversary (Sect. 2.2 for terms) should not be capable of distinguishing between encryptions of 1 and 0, even if the public key is known to the adversary.

Definition 3.4 Let ε be a homomorphic encryption scheme and $\mathcal{A}$ an adversary who has the following advantage with respect to ε:

$$Adv^{\varepsilon}_{\mathcal{A}}(\lambda) \stackrel{def}{=} \Big|Pr\Big[\mathcal{A}(k_p, c) = 1 \mid 1^\tau \leftarrow \mathcal{A}(1^\lambda), \big(k_s, k_p\big) \leftarrow \texttt{KeyGeneration}(1^\lambda, 1^\tau), \\ c \leftarrow \texttt{Encryption}(k_p, 1)\Big] - Pr\Big[\mathcal{A}(k_p, c) = 1 \mid 1^\tau \leftarrow \mathcal{A}(1^\lambda), \\ \big(k_s, k_p\big) \leftarrow \texttt{KeyGeneration}(1^\lambda, 1^\tau), c \leftarrow \texttt{Encryption}(k_p, 0)\Big]\Big| \tag{3.4}$$

The homomorphic encryption scheme ε is called ***semantically secure***, if for all PPT adversary $\mathcal{A}$, the advantage $Adv^{\varepsilon}_{\mathcal{A}}(\lambda)$ is a negligible function depending on λ.

In Definition 3.4, the functionality parameter τ is interpreted as the adversarial threshold. The adversary must maintain its output within this limit to retain its PPT charactertic.

Apart from the standard attributes of encryption schemes, homomorphic schemes possess unique characteristics, which include: the constancy of the secret-key, potent homomorphism, compactness, confidentiality of functions, and multi-hop capabilities. Further, We will define each of them.

Secret-key variant. By examining Definitions 3.2 and 3.3, we can infer the equivalent conditions for the secret key homomorphic encryption: the key generation algorithm yields a single key—the secret key k_s, used for both the encryption and decryption processes; the evaluation function requires the secret key k_s (rather than k_p), alongside π and $\overrightarrow{v}$. When it comes to the definition of semantic security, although there are minor modifications, the fundamental concept remains consistent with Definition 3.4: given the absence of a public key in the secret-key schema, the adversary $\mathcal{A}$ knows a set of encrypted texts (as opposed to being aware of the public key) and the choice of encrypting the bits from one of two possible sequences. The adversary's task is then to identify which sequence was encrypted. Consequently, the advantage of $\mathcal{A}$ will be expressed as follows:

$$
\begin{aligned}
Adv^{\varepsilon}_{\mathcal{A}}(\lambda) = \Bigg| Pr\Big[\mid \overrightarrow{b_0} \mid = \mid \overrightarrow{b_1} \mid \text{ and } \mathcal{A}(pk, \overrightarrow{v} = 1) \Big| \big(1^{\tau}, \overrightarrow{b_0}, \overrightarrow{b_1}\big) \leftarrow \mathcal{A}(1^{\lambda}), \\
k_s \leftarrow \texttt{KeyGeneration}(1^{\lambda}, 1^{\tau}), \overrightarrow{c} \leftarrow \texttt{Encryption}(s_p, \overrightarrow{b_0}) \Big] \\
- Pr\Big[\mid \overrightarrow{b_0} \mid = \mid \overrightarrow{b_1} \mid \text{ and } \mathcal{A}(pk, \overrightarrow{v} = 1) \Big| \big(1^{\tau}, \overrightarrow{b_0}, \overrightarrow{b_1}\big) \leftarrow \mathcal{A}(1^{\lambda}), \\
k_s \leftarrow \texttt{KeyGeneration}(1^{\lambda}, 1^{\tau}), \overrightarrow{c} \leftarrow \texttt{Encryption}(s_p, \overrightarrow{b_1}) \Big] \Bigg|
\end{aligned}
\tag{3.5}
$$

The work in [10] proves the equivalency of definitions between secret key homomorphic encryption and public key homomorphic encryption. As a result, certain constructions only refer to the definitions regarding the secret-key version.

Strongly homomorphic encryption system. This condition, while straightforward, is quite powerful and specific to homomorphic encryption schemes. Strong homomorphism signifies that the evaluated encrypted texts share the same distribution as the freshly encrypted ones. The formal depiction of strong homomorphism is presented as follows:

Definition 3.5 Let ε be a homomorphic encryption scheme and $C = \{C_\tau\}_{\tau \in \mathbb{N}}$ a family of circuits. The homomorphic encryption scheme ε is called ***strong homomorphic over*** C, if $\forall \tau \in \mathbb{N}$ and $\forall \pi \in C_\tau$ and $\forall \overrightarrow{b} = (b_1, \ldots b_t) \in \{0, 1\}^t$, where $\overrightarrow{b}$ is a vector of plain bits, one bit for each entrance of the circuit π, then the following distributions are similar up to a difference negligible in λ:

$$\begin{aligned}
&\texttt{Fresh}_{\pi,\overrightarrow{b}}(\lambda) \overset{def}{=} \left\{(k_p, \overrightarrow{c}, \overrightarrow{c}')\middle|(k_s, k_p) \rightarrow \texttt{KeyGeneration}(1^\lambda, 1^\tau),\right. \\
&\left.\overrightarrow{c} \rightarrow \texttt{Encryption}(k_p, \overrightarrow{b}), \overrightarrow{c}' \rightarrow \texttt{Encryption}\big(k_p, \pi(\overrightarrow{b})\big)\right\} \\
&\texttt{Eval}_{\pi,\overrightarrow{b}}(\lambda) \overset{def}{=} \left\{(k_p, \overrightarrow{c}, \overrightarrow{c}')\middle|(k_s, k_p) \rightarrow \texttt{KeyGeneration}(1^\lambda, 1^\tau),\right. \\
&\left.\overrightarrow{c} \rightarrow \texttt{Encryption}(k_p, \overrightarrow{b}), \overrightarrow{c}' \rightarrow \texttt{Evaluation}\big(k_p, \pi, \overrightarrow{c}\big)\right\}
\end{aligned} \tag{3.6}$$

The definition is readily adaptable to secret key homomorphic encryption. Note that despite the definition's reliance on a family of circuits C, it can be applied to *any* circuit. This is due to the fact that a circuit $C_{\tau^*} \in C$ incorporates a set of gates that offer full functionalities.

While strong homomorphism is a powerful characteristic of homomorphic encryption schemes, it can be so extensive that decryption eventually becomes unfeasible. This is where *compactness* comes into play, absorbing a significant portion of the capabilities of strong homomorphism. Whereas strong homomorphism requires successful decryption of both fresh and evaluated ciphertext, compactness stipulates that the size of the encrypted texts should not expand in tandem with the complexity (depth) of the circuit applied to them.

Definition 3.6 Let ε be a homomorphic encryption scheme and $\overrightarrow{b} = (b_1, \ldots, b_t) \in \{0, 1\}^t$ be a vector of plain bits. The homomorphic encryption scheme ε is called ***compact***, if exists $B(\cdot)$ such that $\forall\lambda, \tau \in \mathbb{N}, \forall\pi$, where $B(\cdot)$ is a fixed polynomial bound and the circuit π has t entrances and one exit, for which the following property is satisfied:

$$\begin{aligned}
&Pr\big[\mid c' \mid \leq B(\lambda) \mid (k_s, k_p) \leftarrow \texttt{KeyGeneration}(1^\lambda, 1^\tau), \overrightarrow{c} \leftarrow \texttt{Encrypt}(k_p, \overrightarrow{b}), \\
&c' \leftarrow \texttt{Evaluate}(k_p, \pi, \overrightarrow{c})\big] = 1
\end{aligned} \tag{3.7}$$

The condition of B depending only on λ implies that the size of the evaluated encrypted texts will not increase even if τ does, where τ can be employed in other components of the scheme (such as in key generation). In a different, less stringent form of compactness, the size of the encrypted texts can increase with τ, but it should not exceed the dimensions of the circuits in C_τ.

The property of *function privacy (circuit privacy)* guarantees that the encrypted texts, derived from the `Evaluation` algorithm, reveal no information about the circuit they were evaluated through, and consequently, do not disclose any details about the function that the circuit represents or the entity that generated the encryption system's keys. If we regard the `Evaluation` algorithm (from Definition 3.2) as a protocol established between the party that generated the encryption system's keys (and encrypted the input values) and the server that will implement a function on the received input (and return the output to the party), then function privacy can be defined as the standard requirement for input privacy expected from a server.

Definition 3.7 Let ε be a homomorphic encryption scheme, C be a family of circuits for which ε is correct and $\overrightarrow{b} = (b_1, \ldots, b_t) \in \{0,1\}^t$ be a vector of plain bits. We say that ε is ***circuit private*** for C if $\exists \mathtt{Sim}$ such that $\forall \tau \in \mathbb{N}, \pi \in C_\tau$, where $\mathtt{Sim}$ represents an efficient simulator, and b_i corresponds to the entrance i of π, for which the following is fulfilled:

$$\mathtt{Real}_{\pi,\overrightarrow{b}}(\lambda) \overset{(c)}{\approx} \mathtt{Sim}(1^\lambda, 1^\tau, \overrightarrow{b}),$$
where
$$\mathtt{Real}_{\pi,\overrightarrow{b}} \overset{def}{=} \{(r, r', \overrightarrow{c}') \mid r, r' \leftarrow \$, (k_s, k_p) \leftarrow \mathtt{KeyGeneration}(1^\lambda, 1^\tau), \overrightarrow{c} \leftarrow \mathtt{Encrypt}(k_p, \overrightarrow{b}),$$
$$\overrightarrow{c}' \leftarrow \mathtt{Evaluate}(k_p, \pi, \overrightarrow{c})\} \tag{3.8}$$

Remark in the Definition 3.7 that $\mathtt{Sim}$ takes as input $\pi(\overrightarrow{b})$, but not π itself. In its behavior, $\mathtt{Sim}$ should simulate the inclusion of the randomness for the $\mathtt{KeyGeneration}$, $\mathtt{Encryption}$, $\mathtt{Evaluation}$. More information about the "efficient simulator" can be found in Lindell [11].

In homomorphic encryption schemes that are not strongly homomorphic, the encrypted text resulting from the $\mathtt{Evaluation}$ algorithm may differ from the freshly encrypted text. This raises the question of whether computations can continue on the evaluated encrypted texts. A homomorphic encryption scheme that allows the $\mathtt{Evaluation}$ algorithm to be applied 'i' times over its own output is termed an ***i-hop homomorphic encryption scheme***. When the homomorphic encryption scheme is *i-hop* for every 'i', it is referred to as a *multi-hop* homomorphic encryption scheme.

3.3 Types of Homomorphic Encryption Schemes

Homomorphic encryption is a specialized form of encryption that allows computations to be performed on encrypted data, generating an encrypted result that, when decrypted, corresponds to the outcome of operations conducted on the original plaintext data. The transformative value of homomorphic encryption is that it allows data to be kept encrypted while undergoing processing or analysis, thereby preserving privacy and security in sensitive contexts. Based on their functional capabilities and limitations, homomorphic encryption schemes can be generally categorized into three types: Partial Homomorphic Encryption (PHE), Somewhat Homomorphic Encryption (SWHE or SHE), and Fully Homomorphic Encryption (FHE).

Partial Homomorphic Encryption (PHE)

Partial Homomorphic Encryption (PHE) represents the earliest class of homomorphic encryption systems. These are encryption schemes which allow specific types of computations to be performed directly on encrypted data. However, PHE systems

are 'partial' in the sense that they support only a single type of operation—either addition or multiplication—but not both.

The Rivest-Shamir-Adleman (RSA) [3], Paillier [12] and ElGamal [4] encryption systems are prime examples of PHE, all of which were not originally designed with the aim of supporting homomorphic properties, yet they inherently possess these capabilities.

The RSA encryption scheme, named after its inventors Ron Rivest, Adi Shamir, and Leonard Adleman, is a PHE system that is *multiplicatively homomorphic*. This implies that the multiplication of two ciphertexts under the RSA encryption will decrypt to the product of the corresponding plaintexts. However, the RSA encryption does not support addition on the ciphertexts. The multiplicative homomorphism of RSA has been utilized in various cryptographic applications such as secure voting and multiparty computation, where operations need to be performed directly on encrypted data.

Below, we will just illustrate the homomorphic property of the RSA encryption algorithm. We encourage the reader to consult [3] for the details of the RSA cryptosystem.

Let $Enc : A \rightarrow A$, $Enc(m) = m^e \mod n$ be the encryption function of the RSA cryptosystem, where e, n are positive integers and $A = \{0, 1, \ldots, n-1\}$. Therefore, the following relations are true:

$$\begin{aligned} Enc(m_1) \cdot Enc(m_2) &= m_1^e \cdot m_2^e \mod n \\ &= (m_1 \cdot m_2)^e \mod n \\ &= Enc(m_1 \cdot m_2) \end{aligned} \tag{3.9}$$

where m_1, m_2 are two plain messages. In Eq. (3.9) it can be easily seen that $Enc(m_1) \cdot Enc(m_2) = Enc(m_1 \cdot m_2)$, which satisfies the homomorphic property given in Definition 3.1.

Another example of a holomorphic cryptosystem is the Paillier cryptosystem [12]. Below we only illustrate the holomorphic properties of the Paillier cryptosystem. We encourage the reader to consult [12] for the details of the Paillier cryptosystem.

- *Additive Homomorphism*: The Paillier cryptosystem supports homomorphic addition of plaintexts. More specifically, the multiplication of two ciphertexts will decrypt to the sum of their corresponding plaintexts. Formally, given two ciphertexts $Enc(m_1)$ and $Enc(m_2)$, we have:

$$Dec(Enc(m_1, r_1) \cdot Enc(m_2, r_2) \mod n^2) = m_1 + m_2 \mod n \tag{3.10}$$

  where m_1, m_2 are two plain messages, r_1, r_2, n are positive integers, $Enc()$ is the encryption function and $Dec()$ is the decryption function. We encourage the reader to consult [12] for the details of the Paillier cryptosystem.
- *Multiplication by a Constant*: The Paillier cryptosystem supports homomorphic multiplication of a plaintext by a constant. That is, raising a ciphertext to the power of a constant will result in the decryption being the product of the original plaintext and the constant. Formally, we have:

$$Dec(Enc(m_1)^c \mod n^2) = m_1 * c \mod n \tag{3.11}$$

where m_1 is a plain message, n is positive integer, c is a constant value, $Enc()$ is the encryption function and $Dec()$ is the decryption function.

On the other hand, the ElGamal encryption system [4], created by Taher Elgamal, is another PHE system with *multiplicative holomorphic* properties, meaning that it allows the multiplication of ciphertexts to correspond to the addition of the plaintexts, and similarly, the addition of ciphertexts corresponds to the multiplication of plaintexts. The ElGamal encryption scheme supports the multiplication of plaintexts by manipulating their corresponding ciphertexts. We encourage the reader to consult [4] for the details of the ElGamal cryptosystem. Given two ciphertexts $Enc(m_1)$ and $Enc(m_1)$, we have:

$$Enc(m_1) * Enc(m_2) = Enc(m_1 * m_2) \tag{3.12}$$

where m_1, m_2 are plain messages, $Enc()$ is the encryption function and $Dec()$ is the decryption function.

Despite their utility in specific scenarios, the scope of PHE systems is largely limited due to their inability to support more than one type of operation. This restricts the range and complexity of computations that can be carried out on the encrypted data. For applications demanding both additive and multiplicative operations on ciphertexts, more advanced forms of homomorphic encryption, such as Somewhat Homomorphic Encryption and Fully Homomorphic Encryption, are required.

However, the concept of PHE laid the foundational groundwork for the development of these more advanced homomorphic encryption systems. The understanding of how to enable computations on encrypted data, even if limited to a single operation, was a key stepping stone to the development of encryption systems that could support more complex operations and eventually arbitrary computations.

Somewhat Homomorphic Encryption (SWHE)

Somewhat Homomorphic Encryption (SWHE) is a significant extension of PHE. While PHE schemes only allow a single type of operation to be performed on the ciphertexts, SWHE systems enable multiple types of operations, typically both addition and multiplication, to be performed on encrypted data. However, there is an important limitation: only a limited number of these operations can be executed before the noise embedded in the encryption grows too large and renders the ciphertext indecipherable upon decryption.

Craig Gentry's pioneering work in 2009, which led to the first fully homomorphic encryption scheme, initially introduced the concept of SWHE. Gentry's construction begins with a somewhat homomorphic encryption scheme that can evaluate low-degree polynomials over encrypted data but becomes too noisy for higher-degree polynomials. He subsequently introduced a "bootstrapping" operation to keep the noise in check and convert the SWHE scheme into a fully homomorphic one.

An example of a somewhat homomorphic encryption scheme is the Brakerski-Vaikuntanathan (BV) scheme [13], based on the learning with errors (LWE) problem. The BV scheme supports both addition and multiplication, but noise grows with each operation and eventually overwhelms the useful signal. This growth in noise restricts the number of operations that can be performed, limiting the computational depth. Here, the problem is that the noise increases so much that, after a certain threshold, the ciphertext obtained from the operations on encrypted data can no longer be decrypted.

Despite these limitations, SWHE schemes are valuable in scenarios where the computational requirements are known and limited. They can securely evaluate functions of a known depth and provide enhanced efficiency compared to fully homomorphic encryption schemes.

SWHE, therefore, acts as a stepping-stone between PHE and Fully Homomorphic Encryption, increasing the range of feasible operations on encrypted data while setting the stage for the introduction of noise management techniques that make FHE possible. Despite the inherent limitations in the computational depth of SHE, its conceptualization and realization have significantly contributed to the evolution of homomorphic encryption and privacy-preserving computations.

Fully Homomorphic Encryption (FHE)

Fully Homomorphic Encryption (FHE) is a game-changer in the realm of cryptography. While PHE and SWHE cryptosystems allow limited operations on encrypted data, FHE enables arbitrary computations on ciphertexts. This means that any function that can be computed on plaintexts can also (or an equivalent function) be computed on their encrypted counterparts, without ever requiring decryption during the computation process. The result of such computations, when decrypted, is identical to the result of the same computations performed on the original plaintext data.

The concept of FHE was first proposed by Rivest, Adleman, and Dertouzos in 1978, with the paper entitled "*On data banks and privacy homomorphisms*" [1] but it was not until 2009 that the first FHE scheme was developed by Craig Gentry [5]. His landmark scheme was based on ideal lattices and introduced the concept of "*bootstrapping*," a technique to keep the noise growth "acceptable", enabling arbitrary computations. Bootstrapping is a critical operation in FHE that allows the scheme to perform an unlimited number of computations on encrypted data.

Fully homomorphic encryption schemes have a property called "noise" associated with encrypted data. As computations on the encrypted data are performed, this noise grows. If it grows too large, it can prevent the successful decryption of the data, essentially making the encrypted data useless. Early versions of FHE could only handle a limited number of computations before the noise became too much and they are categorized as SWHE schemes.

Bootstrapping is a technique introduced by Craig Gentry in his groundbreaking work on FHE, which effectively "refreshes" the ciphertext, reducing the noise to a level where it can be managed, and thus allows for more computations to be done.

More detailed, FHE allows to perform computations on encrypted data, so an encryption of the decryption circuit itself can be created. Then, the encrypted decryp-

tion circuit can be used to "decrypt" the noisy ciphertext while everything remains encrypted. The result is a new ciphertext that represents the same data but with less noise. This is the bootstrapping operation. In other words, bootstrapping allows to take a ciphertext that is close to being undecryptable because of excessive noise and transforms it into a new, less noisy ciphertext that decrypts to the same value. While bootstrapping was a major breakthrough in FHE, it is a computationally expensive operation. As a result, much of the research in FHE is focused on finding ways to reduce the need for bootstrapping or to make it more efficient.

The critical innovation of Gentry's work was demonstrating that it was possible to perform both addition and multiplication operations any number of times on encrypted data without ever revealing it. This was an extraordinary leap forward in terms of data privacy and security, opening new horizons for secure computation, particularly for cloud computing and third-party computation services.

Subsequent FHE schemes have aimed to improve upon Gentry's original construction, reducing computational complexity, increasing efficiency, and exploring new hardness assumptions. Examples include the Brakerski-Gentry-Vaikuntanathan (BGV) scheme [14], the Brakerski-Fan-Vercauteren (BFV) scheme [15, 16], and the Cheon-Kim-Kim-Song (CKKS) scheme [17], which supports approximate arithmetic on encrypted real or complex numbers.

While practical implementation of FHE still poses significant challenges due to its high computational costs, ongoing research is aimed at optimizing these schemes, making FHE a promising tool for the future of secure, privacy-preserving computation in numerous fields including healthcare, finance, machine learning, and more. The pursuit of FHE stands as a testament to cryptography's potential to reconcile the often conflicting goals of utility and privacy in the age of data.

3.4 Fully Homomorphic Encryption

In the digital age, data is a key driver of innovation and economic growth. It fuels everything from online advertising to healthcare research. But, as data has become more valuable, it has also become a prime target for cyberattacks, necessitating robust encryption to ensure data privacy and security. We have seen that FHE cryptographic systems allow computations to be performed on encrypted data without any need for decryption. The implication? Data can remain secure throughout the entire computational process—a game-changing feature with a wealth of potential applications. FHE has potential in a wide range of domains; below, we name a few of them.

Cloud Computing and Data Storage. Cloud computing has become integral to businesses and organizations worldwide, but it carries inherent security risks. Here, FHE can serve as a key line of defense. Businesses can encrypt their sensitive data before uploading it to the cloud. Using FHE, they can perform computations and analyses on this data while it remains encrypted, significantly reducing the risk of data breaches.

Healthcare and Biomedicine. Patient records and biomedical data contain sensitive information that must remain confidential. However, these data also hold valuable insights that can advance medical research and improve patient care. With FHE, it becomes possible to carry out data analyses or machine learning while the data remains encrypted, offering a path to groundbreaking research without infringing on patient privacy.

Machine Learning and Artificial Intelligence. AI and machine learning models often require large amounts of data to learn effectively. With FHE, these models could be trained on encrypted data, enabling organizations to harness the power of their data without exposing sensitive information. This is particularly relevant for industries handling highly sensitive data, such as healthcare, finance, or government.

Secure Voting Systems. In a world increasingly embracing digital solutions, the concept of online voting systems is gathering interest. However, ensuring voter anonymity and data security is a major challenge. FHE can help provide a solution, allowing votes to be counted while ensuring voter privacy and maintaining data integrity.

Financial Services. Banks and financial institutions manage vast amounts of sensitive data. FHE offers the potential to run analyses and algorithms on encrypted data, minimizing the risk of financial fraud and data breaches. This is particularly important in areas such as credit scoring, where personal customer data is used to make financial decisions.

Digital Advertising Personalized digital advertising relies on profiling users based on their browsing habits and preferences. With increasing data privacy regulations, advertisers need to balance personalization with privacy. FHE allows advertisers to analyze encrypted user data to make advertising decisions, without ever accessing the actual personal data.

While FHE holds great promise, it is still a nascent technology, and challenges remain in terms of efficiency and practical implementation. However, ongoing research and advances in the field make it likely that we will soon start to see wider adoption of FHE in these and other domains. As this happens, FHE could become an integral component in the toolkit that enables a secure, privacy-preserving digital future.

3.4.1 Classification

In this section, we classify the FHE schemes based on the underlying hardness assumptions. In general, the classification of the FHE schemes is made based on the generations, in the chronological order of their appearance, but broadly, the classification based on the hardness assumptions behind these encryption schemes generates a grouping similar to the chronological one. In Marcolla et al. [18], the authors have identified four classes of FHE schemes based on the underlying mathematical problems:

1. *Ideal Lattices:* Ideal lattices form the basis of many Fully Homomorphic Encryption (FHE) schemes, including the groundbreaking work by Gentry [5]. These mathematical structures provide a rich framework for constructing encryption schemes that allow for arbitrary computations on encrypted data. However, they also introduce a level of complexity that can make these schemes computationally intensive and challenging to implement in practice.
2. *Integers:* Integer-based FHE schemes, such as the one introduced by Van Dijk et al. [6], offer a simpler and potentially more efficient alternative to ideal lattice-based schemes. By operating directly on integers, these schemes simplify the encryption and decryption processes, potentially making them more practical for real-world applications. In general, the mathematical background for them is the Approximate-Greatest Common Divisor (AGCD) problem, which involves identifying a "near common divisor" given a collection of integers that are close to multiples of a large integer.
3. *LWE and RLWE:* The Learning With Errors (LWE) and Ring Learning With Errors (RLWE) problems form the basis of another class of FHE schemes. These problems are believed to be hard to solve, even for quantum computers, making them a promising basis for post-quantum cryptography. FHE schemes based on LWE and RLWE, such as the one proposed by Brakerski and Vaikuntanathan [13], offer strong security guarantees, but like ideal lattice-based schemes, they can be computationally intensive.
4. *NTRU:* NTRU is a lattice-based cryptographic algorithm that has been used to construct FHE schemes. NTRU-based schemes, such as the one proposed by López-Alt et al. [49], offer a balance between security and efficiency. They are less computationally intensive than some other lattice-based schemes, making them a practical option for many applications. However, they also require careful parameter selection to ensure security.

Further, we will detail the characteristics of each category, mainly following the guidelines and simplified notations used in Marcolla et al. [18].

3.4.1.1 FHE Schemes Based on Ideal Lattices

Ideal latices are the basis for the first FHE scheme proposed by Gentry [5]. Following it, in this category are included schemes that improve Gentry's scheme. We begin by describing Gentry's first work from Gentry [5], based on simplified notations presented in Silverberg et al. [19]. The cryptosystem is based on integer sublattice $\mathcal{L}(I) \subseteq \mathbb{Z}^m$, where m is a positive integer number and I is an ideal. The encryption and decryption algorithms are presented below:

- `Encryption`: Let $m \in \{0, 1\}$ be a single bit message. Based on m, the point $a = m + 2e$ is computed, where $a \in \mathbb{R}^d$ and e is a small error value (random vector having coefficients from $\{-1, 0, +1\}$, for which -1 and $+1$ have the same probability of appearance). To obtain the corresponding ciphertext c, a is translated

into a parallelepiped $\mathcal{P}(key_{public})$. Therefore, $c = a - (\lceil a \cdot key_{public}^{-1} \rfloor \cdot key_{public})$, where $\lceil \cdot \rfloor$ represents operation of rounding to the nearest integer.

- `Decryption`: Based on the ciphertext c, the value a' is computed as follows: $a' = c - (\lceil a \cdot key_{secret}^{-1} \rfloor \cdot key_{secret})$. Here a' represents the translation of c into $\mathcal{P}(key_{secret})$, therefore $a' \mod 2$ will be the plain text obtained from c.

In the scheme from above, key_{public} and key_{secret} represent basis in $\mathcal{L}(I)$. The scheme from above is not bootstrappable because of to the complex nature of the decryption algorithm, which means it cannot be evaluated homomorphically. To rectify this, Gentry proposed that the decryption function of a SHE scheme to be *squashed*. This approach converts the original SHE scheme ϵ, into a different scheme, denoted as $\epsilon*$, that retains the same homomorphic properties but has a simplified decryption function that enables bootstrapping. To make the decryption algorithm less complex, Gentry showed that it is suitable adding to the evaluation key some "additional information" about the secret key [5], in the form of a set of vectors $S = \{s_i | i = 1, \ldots, s\}$, from which is extracted a subset T. Then, key_{secret} is the sum of the components within T and the public extra information is S. In [5], Gentry proves the security of $\epsilon*$, which is based on the SSSP, BDDP, and ISVP.

Following Gentry's work, several other researchers have proposed FHE schemes based on ideal lattices that incorporate batching techniques or squashing procedures. For instance, in Smart and Vercauteren [20], it is proposed a scheme that uses a batching technique to improve the efficiency of the encryption process. This scheme introduced the batching technique, which allows for the encryption of multiple plaintexts in a single ciphertext, thereby improving efficiency. However, despite the batching technique, the scheme still has high computational complexity, which can limit its practicality.

Another work [21] also introduced a squashing procedure in their FHE scheme to reduce the complexity of the decryption circuit. This scheme simplified the squashing procedure, which reduced the complexity of the decryption circuit and improved the overall efficiency of the scheme. While the squashing procedure was simplified, the scheme still requires significant computational resources.

Further refined these techniques, proposing a new FHE scheme that combines both batching and squashing for improved performance are introduced in Scholl and Smart [22]. This work combines both batching and squashing techniques for improved performance, making it more efficient than previous schemes. Still, the scheme still has high computational and storage requirements, making it challenging for practical implementation.

Work [23] also contributed to this line of research, proposing a new FHE scheme based on ideal lattices that incorporates a squashing procedure to enhance security. This scheme incorporates a squashing procedure to enhance security, making it one of the more secure FHE schemes based on ideal lattices. However, the enhanced security comes at the cost of increased computational complexity, which can limit the practicality of the scheme.

FHE schemes based on ideal lattices, starting from Gentry's groundbreaking work to the subsequent schemes, have significantly advanced the field of cryptography, opening up new possibilities for secure computation.

The general strengths of these schemes lie in their theoretical feasibility and the introduction of techniques such as batching and squashing techniques, which have improved the efficiency of encryption and decryption processes. The batching technique, allows for the encryption of multiple plaintexts in a single ciphertext, thereby improving efficiency. The squashing procedure, on the other hand, reduces the complexity of the decryption circuit, further enhancing the overall efficiency of the scheme.

However, despite these advancements, FHE schemes based on ideal lattices also have their drawbacks. The most notable of these is the high computational complexity associated with these schemes. Even with the introduction of batching and squashing techniques, these schemes often require significant computational resources, which can limit their practicality in resource-constrained environments. Additionally, the storage requirements for these schemes can also be high, which can be a challenge for practical implementation.

3.4.1.2 FHE Schemes Based on Integers

FHE based on integers represents another significant development in the field of cryptography. Introduced by van Dijk, Gentry, Halevi, and Vaikuntanathan, this approach offers a conceptually simpler and more efficient method for implementing FHE [6]. The integer-based FHE schemes operate directly on integers, which simplifies the encryption and decryption processes compared to the more complex mathematical structures involved in ideal lattice-based schemes. This simplicity can lead to more efficient computations, making integer-based FHE schemes potentially more practical for real-world applications. Furthermore, integer-based FHE schemes have been shown to offer comparable security to their ideal lattice-based counterparts, making them a compelling alternative for secure computation. In general, integer-based FHE schemes use the AGCD problem. This problem involves identifying the "near common divisor" denoted p, based on a collection of integers $x_0, \ldots, x_n \in \mathbb{Z}$. The integer numbers from the set are randomly selected and are proximate to multiples of the large integer p.

The first FHE scheme of this category is Van Dijk et al. [6], on which we present below:

- `KeyGeneration`: the secret key is an odd integer generated randomly p and the public key is the tuple $(x_0, \ldots, x_n)$, with x_0 odd and $x_0 > x_i$, $\forall i$ and $x_i = pq_i + r$, where q_i, r_i are randomly generated integers.
- `Encryption`: the message $m \in \mathbb{F}_2$ represented as a single bit (therefore, $m \in \{0, 1\}$) is encrypted using the formula $c = (m + 2r + 2\Sigma_{i \in S} x_i)$, with r a randomly generated integer and $S \subseteq \{1, \ldots, n\}$ generated randomly.
- `Decryption`: calculate $(c \mod p) \mod 2$.

This scheme is known for its simplicity and elegance, as it does not require complex mathematical structures like lattices. However, it also has certain drawbacks, such as the need for large ciphertexts and public key. Also, it is vulnerable to Chosen-Ciphertext Attacks (CCA), in which an attacker can choose ciphertexts and learn their corresponding decrypted plaintexts. This ability can potentially be used to reveal information about the secret key. Another attack to which the scheme is vulnerable is Noise Flood Attacks, in which if an attacker can cause the noise to grow too large (a "noise flood"), they can cause the decryption process to fail, leading to a denial of service. In some cases, they might even be able to learn information about the plaintext or the secret key.

Several works have built upon this foundation to improve the efficiency and security of integer-based FHE. For example, in Coron et al. [24] proposed a variant of the original scheme with smaller ciphertexts. Similarly, in Cheon et al. [25] is introduced a new technique called *batching* to reduce the noise growth in the encryption process, thereby enhancing the efficiency of the scheme. *Batching* is a technique that enables simultaneous encryption and processing of multiple plaintexts within a single ciphertext, which leverages the Chinese Remainder Theorem (CRT) to pack a vector of plaintexts into a single ciphertext. This means that a set of plaintext messages can be encrypted together in one go, and homomorphic operations can then be performed on all these messages at the same time. This is particularly useful in terms of computational efficiency when performing operations on large datasets, as it reduces the computational overhead associated with encrypting and processing each plaintext individually. However, while *batching* improves computational efficiency, it may also introduce additional complexity in terms of managing the packed ciphertexts and ensuring that the correct operations are applied to the correct plaintexts within the batch. Moreover, the security implications of *batching* are also an important consideration, as any vulnerabilities in the batching process could potentially impact all plaintexts within a batch.

Despite these advancements, integer-based FHE still faces challenges, particularly in terms of security. For example, in Albrecht et al. [26] is proved that these schemes are vulnerable to a variant of the so-called "approximate greatest common divisor" attack. This highlights the need for further research to address these security issues.

3.4.1.3 FHE Schemes Based on LWE and RLWE

Another category of FHE schemes is based on LWE and RLWE problems, which have emerged as a prominent area of research in modern cryptography. These problems, which are believed to be hard to solve even for quantum computers, form the foundation of several secure and efficient FHE schemes. LWE-based and RLWE-based FHE schemes offer a combination between security and functionality and, despite their complexity, these schemes have been shown to be practical for many real-world applications, thanks to ongoing advancements in algorithm design and hardware acceleration. However, like all cryptographic systems, they also face challenges, particularly in terms of computational efficiency and resistance to certain

types of attacks. As such, the development and analysis of LWE-based and RLWE-based FHE schemes remain an active and important area of research in cryptography.

The LWE and RLWE techniques are so important in FHE, that they have generated three generations of FHE schemes: the first generation is constructed around Brakerski-Vaikuntanathan (BV) [13] and Brakerski-Gentry-Vaikuntanathan (BGV) [14] schemes, the second generation is constructed around Gentry-Sahai-Waters (GSW) [27] scheme and the last generation is constructed around Cheon-Kim-Kim-Song (CKKS) [17] scheme.

Although initially, LWE and RLWE were used in the bootstrap operation in Brakerski and Vaikuntanathan [13, 28] respectively that improve Gentry's scheme, the first FHE scheme based on LWE is Brakerski and Vaikuntanathan [13], resulting in the **first generation of (R)LWE-based FHE schemes**. Below, we present the LWE-based FHE scheme (known as the BV scheme) from Brakerski and Vaikuntanathan [13]:

- `Encryption`: Let $m \in \mathbb{F}_2$ be the message. To obtain the ciphertext, the following formula is used: $c = (a, b = \langle a, s \rangle + 2e + m) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, with e being a small error chosen randomly from a distribution error ξ, $s \in \mathbb{Z}_q$ is the secret key formed from a tuple of randomly generated integers in $\mathbb{Z}_q$.
- `Decryption`: To decrypt the ciphertext, the following formula is used: $(b - \langle a, s \rangle \mod q) \mod 2$. This is equvalient with $(2e + m \mod q) \mod 2$. In order to decrypt the message, e should be small enough, i.e. $e < q/2$.

In Brakerski and Vaikuntanathan [13] the authors introduced the technique called *re-linearization*, which is used to manage the "noise" that accumulates during homomorphic operations, particularly multiplication. This noise, if not properly managed, can grow to a point where it makes decryption impossible. The *re-linearization* step control this noise growth, being used after each homomorphic multiplication operation to reduce the degree of the resulting ciphertext back to one, which in turn helps to control the noise. The re-linearization process involves the use of evaluation keys. These evaluation keys are essentially encryptions of the secret key raised to different powers, and they are used during the re-linearization process to transform the multiplied ciphertexts back to their original form, thereby reducing the noise.

Another technique introduced in Brakerski and Vaikuntanathan [13] is the *dimension modulus reduction* used to control the growth of noise during homomorphic operations. In this scheme, the ciphertexts are represented as vectors in a high-dimensional space, and the noise is associated with the length of these vectors. As homomorphic operations are performed, the dimension of the vectors (and thus the amount of noise) can increase, which can eventually lead to correct decryption. The *dimension modulus reduction* technique is used to reduce both the dimension of the vectors and the modulus of the ciphertexts, which in turn reduces the amount of noise. This is done in such a way that the underlying plaintexts are not affected, allowing the correct decryption of the results of homomorphic operations.

An improved version of the scheme proposed in Brakerski and Vaikuntanathan [13] is presented in Brakerski and Vaikuntanathan [28], for which the underlying problem is the polynomial LWE problem, equivalent to RLWE problem [29]. The

scheme [28] works the same as the one from Brakerski and Vaikuntanathan [13], but its component (keys, plain message and encrypted text) are elements within a ring R defined as $R = \mathbb{Z}_q[x]/\langle f(x)\rangle$, where q is a prime number, and $f \in \mathbb{Z}[x]$ has a degree d.

Another scheme based on LWE is proposed in Brakerski et al. [14], where the authors call the scheme *leveled* FHE scheme. This scheme from Brakerski et al. [14] is another important FHE scheme, namely BGV scheme, that is implemented in several libraries used in real-world applications (see Sect. 3.4.4). A *leveled* scheme is a type of FHE scheme that supports a limited number of homomorphic operations. In other words, it can handle computations of a certain "depth" or "level" in the associated circuit. This depth is fixed when the encryption scheme is set up, and it determines the maximum complexity of computations that can be performed on encrypted data. In a leveled FHE scheme, the complexity of the encryption and decryption processes, as well as the size of the keys and ciphertexts, grows with the maximum depth of the computations. This is in contrast to a "pure" or "bootstrappable" FHE scheme, which can handle computations of arbitrary depth, but at the cost of more complex encryption and decryption processes. Leveled FHE schemes are often more efficient than bootstrappable FHE schemes for applications where the maximum depth of the computations is known in advance and is not too large. They have been used in a variety of applications, including secure data analysis, privacy-preserving machine learning, and secure multi-party computation. Being one of the most important FHE schemes, we present below its details according to Brakerski et al. [14]:

- `Setup` algorithm provides the elements that will be used in the subsequent algorithms: $d \in \mathbb{N}*$ is a power of 2, q is an odd positive integer, ξ is an error distribution over the ring $R = \mathbb{Z}[x]/\langle x^d + 1\rangle$, B is a bound on the length of elements within ξ. The dimension of B should be minimum such that the security to be preserved. The last element is the ring $R_p = \mathbb{Z}_p[x]/\langle x^d + 1\rangle$, for any $p \in \mathbb{N}$.
- `KeyGeneration`: here, the security parameter is λ; an element $s \in \xi$ is randomly chosen such that it is a small value, the value $\mathtt{s} = (1, s) \in {R_p}^2$ is computed, then $a' \in R_p$ is randomly generated, and $b = a's + 2e$ is computed based on the small error $e \in \xi$. The secret key is $key_{secret} = \mathtt{s} = (1, s)$ and the public key is $key_{public} = \mathtt{a} = (b, -a)$. As a remark, $\langle \mathtt{a}, \mathtt{s}\rangle = 2e$.
- `Encryption`: This algorithm encrypts the plan message $m \in R_2$ using key_{public}, as follows: set $\mathtt{m} = (m, 0) \in R_2$ and randomly choose $r, e_0, e_1 \in \xi$. The ciphertext is $\mathtt{c} = \mathtt{m} + 2(e_0, e_1) + \mathtt{a}r$. Therefore, $\mathtt{c} = (c_0, c_1) = (m + 2e_0 + br, 2e_1 - a'r) \in {R_q}^2$.
- `Decryption`: The decryption algorithm computes the following value $\langle \mathtt{c}, \mathtt{s}\rangle = c_0 + c_1 s = m + 2e_0 + 2e_1 s + 2er$. The result is $((m + 2(e_0 + e_1 s + er)) \mod 1) \mod 2 = m$.

A remark on the decryption algorithm is that it works because the elements $e, e_0, e_1 \in \xi$ are small enough. The homomorphic properties of the scheme from above are as follows:

- Addition: This is the simplest operation of the two, as it works component by component (component-wise). As long as the error does not extend over modulus q, the decryption works properly.
- Multiplication: This operation is not as natural as addition. In order to "see" the homomorphism, an important remark is made here: $\langle \mathtt{c}, \mathtt{s} \rangle \cdot \langle \mathtt{c}', \mathtt{s} \rangle = (c_0 + c_1 s)(c_0' + c_1' s) = c_0 c_0' + s(c_0 c_1' + c_1 c_0') + c_1 c_1' s^2 = d_0 + d_1 s + d_2 s^2$. That is, an *extended* secret key $(1, s, s^2)$ can be used to decrypt the *extended* ciphertext (d_0, d_1, d_2). The drawback is that each multiplication increases the secret key. To avoid this, in Brakerski et al. [14] have used a method called *switching key*. Shortly, the term $d_2 s^2$ is converted to $\overline{c}_0 + \overline{c}_1 s$ based on the corresponding encryption of s^2 under s. Therefore, the encryption becomes:

$$\mathtt{Encryption}(s^2) = (\beta, -\alpha), \tag{3.13}$$

where

$$(\beta, -\alpha) = (s^2 + 2e + a'rs, 2e_1 - a'r) \approx (s^2 + \alpha s, -\alpha). \tag{3.14}$$

Therefore, $s^2 \approx \beta - \alpha s$, while the *extended* ciphertext $d_0 + d_1 s + d_2 s^2$ becomes a *regular* ciphertext $\overline{c}_0 + \overline{c}_1 s$ that encrypts the same plaintext.

Based on the scheme from above, a leveled version was constructed in the same work [30].

Other several LWE and RLWE are constructed based on or are improvements of the schemes presented in Brakerski and Vaikuntanathan [13] or Brakerski et al. [14]. An example is Gentry et al. [31] which improves the BGV scheme. The key improvement introduced in the paper is related to the bootstrapping technique, which refreshes ciphertexts and removes noise in order to enable continued homomorphic evaluation. The authors introduce techniques to reduce the computational cost of bootstrapping, making it more efficient and practical for evaluating the AES circuit. These optimizations focus on minimizing the number of required modulus switches and reducing the overall complexity of the bootstrapping operation. Another contribution is noise reduction based on methods to effectively manage and reduce noise during homomorphic evaluation of the AES circuit. By carefully controlling and manipulating the noise levels, the efficiency and accuracy of the bootstrapping operation are improved. Lastly, the overall performance is improved and the authors demonstrate that their techniques enable more efficient and faster computations, reducing the overall computational overhead associated with homomorphic operations on the AES circuit. This improvement makes the evaluation of the AES circuit more practical and feasible within the BGV scheme.

The work [32] improves the efficiency and controls the error growth more effectively from Brakerski and Vaikuntanathan [13]. The key idea behind their improvement is to perform the bootstrapping operation on a ciphertext encrypted under a smaller dimension, which reduces the computational overhead of the bootstrapping operation. The authors achieve this by using a "ring-switching" procedure to change

the ring over which the ciphertext is defined, effectively reducing its dimension. Furthermore, they introduce a new technique to control the error growth during the bootstrapping procedure. The authors' new bootstrapping procedure uses a "modulus-switching" technique to effectively reduce the error in the ciphertext, which improves the accuracy of the computations performed on the encrypted data.

Other works constructed around BV or BGV are Fan and Vercauteren [16] that enables arithmetic operations made directly on the encrypted data, Bajard et al. [33] is an improvement of Fan and Vercauteren [16] that uses Residue Number System and Chinese Remainder Theorem when the coefficients are too large, Halevi et al. [34] is an improvement of Bajard et al. [33], but in Bajard et al. [35] is shown that actually the performance for the Bajard et al. [33], Halevi et al. [34] are similar. Lastly, in Chen and Han [36] the authors improved the bootstrapping algorithm for both BV and BVG. Note that besides the works mentioned in this section, there are many others centered around BV and BVG and their variants.

The second generation of LWR and RLWE-based FHE schemes is centered around the GSW scheme [27]. Below, we present a simpler version of the GSW scheme [37] that is not multiplication homomorphically based on the guidelines from Alperin-Sheriff and Peikert [32]:

- `KeyGeneration`: the secret key is the tuple $\mathtt{s} = (1, s_2, \ldots, s_n) \in {\mathbb{Z}_p}^n$, with values s_i randomly chosen. The public key is an $n \times n$ matrix $A \in {\mathbb{Z}_p}^{n \times n}$ with the property that $A \cdot \mathtt{s} = \mathtt{e} \approx 0$.
- `Encryption`: the ciphertext is computed using the formula $C = mI_n + RA$, with the message $m \in \mathbb{Z}_q$, the identity matrix I_n, the randomly generated $n \times n$ matrix R with elements in $\mathbb{F}_2$. These settings ensure that the entries within R are small.
- `Decryption`: to decrypt the encrypted message, firstly, the value $C\mathtt{s} = mI_n + RA\mathtt{s} = mI_n + R\mathtt{e} \approx mI_n s$ is computed. Based on the fact that R is small it results that if $A\mathtt{s} \approx 0$ then $RA\mathtt{s} \approx 0$. Then, the vector x is computed as $x \approx mI_n\mathtt{s} \approx (ms_1, \ldots, ms_n)$, from which it results m (because $s_1 = 1$).

It is employed to approximate the eigenvector associated with the decryption operation, allowing for more efficient decryption of ciphertexts.

As we have seen in the above GSW scheme, the encryption process involves transforming plaintext messages into ciphertexts using a lattice-based encryption mechanism. However, in the FHE version of GSW [27], to decrypt a ciphertext, the authors introduced a technique called the *approximate eigenvector method*. In the FHE GSW, during the decryption process, the approximate eigenvector method is used, as follows: to decrypt a ciphertext, the encrypted matrix is multiplied by an approximate eigenvector. The eigenvector is approximate because exact computation would be computationally expensive. The multiplication with the approximate eigenvector effectively "clears" the noise in the ciphertext, allowing for an approximate recovery of the original plaintext. After multiplying the ciphertext by the approximate eigenvector, the result is a new ciphertext that has reduced noise. This reduced-noise ciphertext can be transformed back into an approximate version of the original plaintext.

Even though GSW is important, it has a few drawbacks. It produces ciphertexts with a large size compared to the size of the original plaintext. This expansion in ciphertext size results in increased storage requirements and communication overhead, making the scheme less efficient in practice. Also, it has relatively high computational costs for both encryption and decryption operations. The encryption process involves extensive matrix operations, and the decryption process requires the multiplication of ciphertexts by approximate eigenvectors, which can be computationally expensive, leading to slower computation times and lower overall efficiency. The GSW scheme is susceptible to noise growth during homomorphic operations. As computations are performed on ciphertexts, noise accumulates and can affect the correctness and security of the scheme. Lastly, the GSW scheme is primarily designed for evaluating arithmetic circuits, making it less flexible compared to more recent FHE schemes. It may not support certain types of computations, such as complex Boolean operations or non-arithmetic functions, without additional transformations or modifications. However, even though GSW has these drawbacks, it is the basis for other improved FHE schemes.

In Ducas and Micciancio [38] is presented FHEW encryption scheme, an implementation of the GSW scheme that can perform bootstrapping in less than a second. This scheme achieved significant improvements in the depth reduction process. By reducing the number of levels required to evaluate circuits, the scheme from Ducas and Micciancio [38] minimized the computational overhead and improved the overall efficiency of homomorphic computations. Moreover, it uses an optimized version of the bootstrapping operation based on a technique that "refreshes" ciphertexts and eliminates noise accumulation during homomorphic operations. This bootstrapping operation is more efficient than previous approaches. Based on the equality tests, the evaluation of equality circuits is made homomorphically. This improvement allowed for more complex computations involving comparisons and conditional operations, expanding the range of applications that could be efficiently handled using the FHEW scheme. Also, the scheme from Ducas and Micciancio [38] introduced techniques to reduce the memory requirements of the scheme, making it more practical and feasible for real-world implementations. By optimizing the storage of certain components, FHEW achieved a more efficient utilization of memory resources.

Another improvement is presented in Chillotti et al. [39], which actually improves [38]. Here, it is introduced TFHE, in which authors propose a new method for performing packed homomorphic operations, which significantly improves the efficiency of the TFHE scheme.

Other schemes constructed around GWS are [40–42], but note that these are just a few examples, existing much more schemes constructed around GSW.

The **third** and most recent **generation of (R)LWE-based FHE** schemes are constructed around CKKS [17]. Initially, in Cheon et al. [17] the authors proposed a leveled homomorphic encryption scheme which was known as HE for Arithmetic of Approximate Numbers (HEEAN), therefore the evaluation of the circuit can be made until a certain depth. However, in [43], the same authors presented an improved FHE version of their scheme. Below, we present the leveled version from [17]:

- `Setup`: Let $d = 2^M$ be an integer value and define the ring $R = \mathbb{Z}[x]/\langle x^d + 1\rangle$. Let p be a base, q_0 a module, and L a positive integer and set $q_l = p^l \cdot q_0, l = 1, \ldots, L$. Further, the following distributions are important: $DG(\sigma^2)$ is a Gaussian distribution over $\mathbb{Z}^d$, where $\sigma \in \mathbb{R}$ and σ^2 is the variance; $ZO(\rho)$ is a distribution defined on $\{-1, 0, +1\}^d$ that picks 0 with the probability $1 - \rho$, and picks 1 or -1 with the probability $\rho/2$. Lastly, let ξ be a B-bunded distribution. A remark is that the ciphertext of depth l is a vector in R_{q_l}.
- `KeyGeneration`: Based on the security parameter λ, keys are generated as follows: choose integer values t, h, choose M, choose $\rho \in \mathbb{R}$. When ρ is chosen, take into consideration that the complexity of the attack against RLWE to be 2^λ. The secret key $\mathtt{sk} = (1, s)$, with $sin\xi$. Generate a value $a \in R_{q_L}$ and compute $-as + e \mod q_L$, with $e \in DG(\rho^2)$. Get the samples $a' \in R_{t \cdot q_L}$ and $e' \in DG(\rho^2)$ and calculate the value $b' = -as + e' + ts' \mod (t \cdot q_L)$. The secret key remains $\mathtt{sk} = (1, s)$, the public key is $\mathtt{pk} = (b, a) \in R^2_{q_L}$, and the evaluation key is $evk = (b', a')$.
- `Encryption`: The plain message is $m \in R$. Choose randomly the following values: $v \in ZO(1/2), e_0, e_1 \in DG(\rho^2)$. The ciphertext is computed as $\mathtt{c} = (\beta, \alpha) = v \cdot \mathtt{pk} + (m + e_0, e_1) \mod q_l \in R^2_{q_l}$.
- `Decryption`: Computes the value $m = \langle \mathtt{sk}, \mathtt{c} \rangle \mod q_l = \beta + \alpha \cdot s \mod q_l$.

The homomorphic properties of the scheme presented above are:

- Addition: this operation is trivial and involves the addition of two ciphertexts: $\mathtt{c}_1 + \mathtt{c}_2$.
- Multiplication: this is a little more complex. Let $\mathtt{c}_1 = (\beta_1, \alpha_1)$ and $\mathtt{c}_2 = (\beta_2, \alpha_2)$. Then

$$\mathtt{c}_1 \cdot \mathtt{c}_2 = (d_0, d_1) + \left\lfloor t^{-1} \cdot d_2 \cdot \mathtt{evk} \right\rceil \mod q_l, \tag{3.15}$$

where $(d_0, d_1, d_2) = (\beta_1\beta_2, \alpha_1\beta_2 + \alpha_2\beta_1, \alpha_1\alpha_2) \mod q_l$.

If there are situations in which there should be evaluated two ciphertexts with different levels $l_1 < l_2$, then the ciphertext with the higher level should be reduced to the level of the other ciphertext. This operation is called *rescaling* and converts a ciphertext $\mathtt{c}_2 \in R^2_{q_{l_2}}$ into a ciphertext of level l_1 by computing $\mathtt{c}_1 = \left\lfloor q_{l_2}/q_{l_1} \right\rceil \mod q_{l_2}$.

Compared to its predecessors, the CKKS scheme presents some advantages:

- Complex Numbers: Unlike many other FHE schemes, CKKS supports the encryption and computation of complex numbers, not just integers. This makes it suitable for a wider range of applications, including those involving signal processing, machine learning, and scientific computation.
- Efficiency: CKKS is designed to be more efficient than other FHE schemes, particularly when dealing with approximate numbers (real or complex). This efficiency makes it more practical for real-world use.
- Noise Management: The bootstrapping operation is a bit more complex due to the scheme's support for real and complex numbers, but it is based on *modulus switching*, which changes the ciphertext modulus and scales down the noise. This is

followed by a homomorphic approximation of the complex exponential function, which is used to rotate the encrypted complex number back to the real axis.
- Privacy-Preserving Machine Learning: CKKS is particularly well-suited for privacy-preserving machine learning applications. Machine learning often involves computations on real numbers, which CKKS supports. By using CKKS, a machine learning model can be trained and make predictions on encrypted data, preserving the privacy of the data.

An immediate improvement of Cheon et al. [17] is [43], proposed by the same authors, that consists in *ciphertext packing technique*. This is a method used to encode multiple plaintext values into a single ciphertext and it is useful because it enables efficient computation on multiple data points simultaneously, which significantly improves the performance of homomorphic operations. In the context of the CKKS scheme, the ciphertext packing technique is based on the Chinese Remainder Theorem and it is used to encode complex numbers into a single ciphertext. This is done by viewing the plaintext space as a polynomial ring and encoding the complex numbers as coefficients of a polynomial. This enables efficient computation on vectors of complex numbers. The ciphertext packing technique is also important for the bootstrapping operation in the CKKS scheme. By using the ciphertext packing technique, the bootstrapping operation can be performed on multiple data points simultaneously, significantly improving the efficiency of the process. Other following improvements especially for the packing technique are Kim and Song [44], Kim et al. [45], Boemer et al. [46]. For example, in Kim et al. [45], the authors proposed a technique that refreshes the ciphertext before multiplication, instead of after multiplication. However, at the moment of writing this book, the CKKS scheme is very popular and many improvements have been proposed besides the one mentioned in this section.

We have seen in this section that (R)LWE-based FHE schemes have become a powerful tool in the context of secure computation, offering a robust foundation for privacy-preserving operations on encrypted data. These schemes, such as BGV, FV, GSW, and CKKS, have demonstrated their strengths in various applications, including secure cloud computing, privacy-preserving machine learning, and secure data analysis, among others.

One of the key strengths of (R)LWE-based FHE schemes is their strong security guarantees, which are based on the hardness of the (R)LWE problem, a problem that is believed to be hard even for quantum computers. This makes these schemes a promising choice for post-quantum cryptography. Furthermore, techniques like ciphertext packing and bootstrapping have significantly improved the efficiency of these schemes, enabling operations on multiple data points simultaneously and allowing for an unlimited number of homomorphic operations, respectively.

However, despite these strengths, (R)LWE-based FHE schemes also face several challenges. The most significant of these is the computational and storage overhead associated with these schemes. FHE operations, especially bootstrapping, are computationally intensive and the ciphertexts produced by these schemes are significantly larger than the original plaintexts. This makes these schemes impractical for use in

resource-constrained environments. Additionally, managing the noise growth during homomorphic operations is an important aspect of these schemes. While techniques like bootstrapping can help control the noise growth, they come with a significant computational cost.

3.4.1.4 FHE Based on NTRU

FHE schemes based on the NTRU cryptosystem [47, 48] represent an important class of cryptographic protocols designed to enable computation on encrypted data. These schemes have gained significant attention due to their potential to provide robust security and privacy guarantees in a variety of applications, ranging from secure cloud computing to privacy-preserving machine learning and data analysis.

The NTRU cryptosystem is a lattice-based cryptographic system that has been recognized for its efficiency and strong security properties. Unlike many other cryptographic systems, NTRU is resistant to attacks by quantum computers, making it a promising choice for post-quantum cryptography. FHE schemes based on NTRU use the properties of the NTRU lattice to enable homomorphic operations on encrypted data.

In this section, we will explore the main NTRU-based FHE schemes, and discuss their strengths and limitations.

The first NTRU-based FHE scheme is called LTV (Lopez-Tromer-Vaikuntanathan) and is introduced in López-Alt et al. [49]. Here, the authors used bootstrapping and modulus switching methods in order to eficientize the scheme. The mathematical problem that is behind [49] is Decisional Small Polynomial Ration (DSPR). It works as follows [49]:

- `KeyGeneration`: in order to generate the key, there are used two small polynomials randomly generated $f', g \in \xi$, with ξ being a B-bounded distribution over the ring $R = \mathbb{Z}[x]/\langle x^d + 1\rangle$ with $d \in \mathbb{N}$ a power of 2. The secret key is $f = 2f' + 1 \in R$, with $f \equiv 1 \mod 2$ and f invertible in R. The public key is $h = 2gf^{-1} \mod q \in R$.
- `Encryption`: a message is a value $m \in \mathbb{F}_2$ and it is encrypted using the formula $c = hs + 2e + m \in R_q = \mathbb{Z}_q[x]/\langle x^d + 1\rangle$, with s, e randomly chosen from ξ.
- `Decryption`: the plain message is recovered as $m = (fc \mod q) \mod 2$.

In this context, the DSPR problem is based on the fact that h from the scheme cannot be distinguished by any polynomial in R_q. Subsequent work based on LTV is Bos et al. [50], in which the authors proposed two versions of SHE, one for which DSPR is replaced with the scale invariant and the second one being an improvement for the first. An interesting approach is presented in Doróz and Sunar [51], where the authors proposed a GWS-based scheme, but with NTRU properties. In general, NTRU schemes are faster than regular RLWE schemes, but depending on the dimension of the plaintext it can be vice-versa [52]. Also, parameters for NTRU schemes are important in order to avoid certain attacks, as proposed in Cheon et al. [53].

NTRU-based FHE schemes have been shown to produce short ciphertexts, which can significantly reduce storage and communication overhead. They have also been designed to be highly efficient, enabling complex computations on encrypted data, a feature that is particularly beneficial for secure outsourced computation and cloud computing applications.

However, there are several challenges to these schemes. The increased efficiency often comes with increased complexity, which can make these schemes difficult to implement and use in practice. Furthermore, while these schemes provide strong theoretical security guarantees, thorough security analyses and performance evaluations are needed to fully understand their resistance to various types of cryptographic attacks and their performance under heavy workloads or in large-scale deployments.

3.4.2 Standardization

With applications ranging from secure cloud computing to privacy-preserving AI, FHE is indeed a technological "holy grail". However, for FHE to become truly mainstream and universally accepted, we need something more than just technical brilliance: standardization.

Standardization is the process of developing and implementing technical standards based on the consensus of different parties, including companies, users, interest groups, standards organizations, and governments. The goal is to ensure that a certain technology performs consistently and as expected, regardless of the user, provider, or geographical location.

In the context of FHE, standardization would lead to several benefits:

- *Interoperability*: Standards ensure that different FHE systems can work together seamlessly, facilitating widespread adoption.
- *Security Assurance*: Standards would define a baseline for security, enabling users to trust that their data is safe when encrypted with FHE.
- *Encourage Innovation*: By defining core requirements and best practices, standards enable companies to focus on creating innovative solutions around these parameters.

Standardization is essential to the adoption of any technology. It provides a common language that enables interoperability between different systems, promotes confidence in the technology, and facilitates its use. In the context of FHE, standardization is even more critical given the sensitivity and privacy of the data being handled. There are several initiatives underway to standardize FHE.

3.4.2.1 HomomorphicEncryption.org

HomomorphicEncryption.org is a dedicated community of researchers and practitioners in the field of cryptography. It serves as a hub for knowledge sharing, collaboration, and standardization for HE technologies. Their primary goal is to develop

and maintain standards for the implementation of HE to ensure interoperability and reliability in real-world applications. By working to develop these standards, the organization is promoting a future where encrypted data can be securely processed without ever being exposed.

The standardization work of the community revolves around three main cryptographic schemes: BGV, BFV, and CKKS mentioned in Sect. 3.3. These schemes represent the most widely accepted implementations of FHE and provide a basis for compatibility and consistency across multiple platforms and applications. By standardizing these schemes, HomomorphicEncryption.org is laying the foundation for a safer digital ecosystem.

One of the main initiatives of HomomorphicEncryption.org is the development of the Homomorphic Encryption Standard (HES). This standard provides guidelines for implementing and using HE schemes. Additionally, the organization hosts annual workshops where experts gather to share their research, discuss developments, and push forward the standardization process.

Moreover, HomomorphicEncryption.org launched an early version of the *Security Standard* in 2018 [54], which sets the security parameters for the aforementioned three schemes, ensuring that the implementation of FHE offers the highest possible level of security.

The work of HomomorphicEncryption.org has wide-reaching implications for multiple industries. With the standardization of FHE, sectors like healthcare, finance, government, and many others can safely analyze encrypted data, gain insights, and make critical decisions without compromising on data privacy.

3.4.2.2 NIST Post-quantum Cryptography

The National Institute of Standards and Technology (NIST) has played an important role in cryptographic standards, notably in the area of post-quantum cryptography (PQC) [55]. Post-quantum cryptography is a subset of cryptography that aims to develop cryptographic systems that are secure against both quantum and classical computers. This has become an increasingly urgent priority with the advent of quantum computing, which poses a potential threat to classical cryptographic schemes.

NIST is running a competition-style project to standardize post-quantum cryptographic algorithms and this has great implications for FHE. FHE algorithms are not inherently resistant to quantum attacks. Therefore, the progress and final results of NIST's post-quantum standardization process can significantly influence the future design and acceptance of FHE schemes. A post-quantum secure FHE would mean that computations on encrypted data could resist potential quantum threats.

By examining proposed algorithms and testing them against rigorous standards, NIST is leading the way in establishing a set of cryptographic algorithms that are resistant to quantum attacks. These algorithms are expected to be utilized in various encryption schemes, including FHE, enhancing their security profiles and making them viable in a post-quantum world.

3.4.2.3 ISO/IEC WD 18033-8

International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) working draft (WD) 18033-8 [56] outlines cryptographic procedures for calculating a function on encrypted data, maintaining the secrecy of the input, intermediate data, and the output during computation using FHE. The document establishes definitions, symbols, and format requirements. It details the security model, underlying assumptions, spaces for messages, ciphertexts, and keys, along with their formats. Additionally, it prescribes cryptographic techniques for the cryptographic schemes that are apt for standardization.

In April 2021, the ISO/IEC Joint Technical Committee 1 and 2 unanimously agreed to propose a new work item to extend the existing series of encryption standards ISO/IEC 18033, with a Part 8 dedicated to Fully Homomorphic Encryption.

The experts evaluated the need for fully homomorphic encryption across both public and private sectors, including healthcare, retail, and finance sectors among others. They highlighted several important use cases that can significantly benefit society, such as the enhanced credit system, Covid tracing, or stolen password detection.

The security parameters proposed by HomomorphicEncryption.org were considered a suitable starting point, but the experts also highlighted the necessity for further refinement by introducing security margins to cover both classic and quantum-safe security levels. The experts also emphasized the inclusion of security notions of *Ciphertext Indistinguishability* (IND-CPA) in a forthcoming standard.

The experts agreed that the schemes BGV, BFV, CKKS mentioned in Sect. 3.3, and CGGI (TFHE) [57] meet the maturity criteria for standardization under ISO/IEC. Furthermore, they also agreed that the specifications should be comprehensible enough for implementation and contain all the necessary details to ensure interoperability. The experts also outlined a provisional table of contents for a potential standard, drawing references from the established literature to guide the writing of the specifications of the schemes.

3.4.3 Open Source Libraries

Although these are not really standards, the open-source libraries that implement FHE schemes are worth being mentioned in this section. For example, open-source libraries such as Microsoft's SEAL, IBM's HElib, and PALISADE have established de facto standards for FHE implementations. These libraries reflect the current best practices and often serve as reference implementations.

While these efforts represent significant strides towards standardization, there is still much work to be done. With the collaborative efforts of all stakeholders, we can hope to see FHE standards established that promote a future where data can be both useful and private—a balance that is increasingly critical in our data-driven world. However, at the moment of writing this book, the status of the standards discussed in this section is under development.

3.4.4 Implementations

Implementations of homomorphic encryption in the form of libraries have an important role in cryptography, mostly in promoting usability, accessibility, and increasing the use cases for homomorphic encryption schemes in real-world applications.

However, developing a homomorphic encryption system from scratch is a complex process that requires strong knowledge. Having these functionalities readily available in libraries simplifies the integration process, making these robust cryptographic techniques more accessible to developers who may not specialize in cryptography. This helps to a broader usage of homomorphic encryption, letting its benefits be used in more diverse applications. Moreover, these libraries can offer a high level of security. Usually, they are typically developed and maintained by experts in the field who understand the nuances and pitfalls of implementing cryptographic systems. The libraries are often open-source, a fact that allows for continuous examination, testing, and improvement of the code by the cryptographic community. Involving experts from the entire community is an advantage because the likelihood of possible errors that could compromise the security of the system is eliminated.

These libraries are also important for the FHE standardization process. Based on the standardization facilitated by libraries interoperability between different systems and platforms becomes easier to achieve. The ability to have different systems '*speaking the same language*' is one of the foundations of modern interconnected systems. Moreover, well-developed libraries typically come with thorough documentation and examples, which makes it easier for developers to understand and correctly use the cryptographic primitives, reducing the danger of misuse and consequential security vulnerabilities.

In essence, homomorphic encryption libraries are important in bridging the gap between theoretical cryptographic advancements and their practical, real-world applications. Below, we provide a list of implementations of homomorphic encryption schemes, according to Homomorphic Encryption Standardization [58], Gouert et al. [59], Doan et al. [60]:

- **Microsoft SEAL (C++, .NET)**[1]: This library is used to perform computations directly on encrypted data, known as homomorphic encryption. It supports both the BFV and CKKS schemes, allowing arithmetic operations to be performed on encrypted integers or real/complex numbers, respectively. It has a high-level API that allows developers to work with encrypted data without needing to understand the underlying mathematics.
- **OpenFHE(C++)**[2]: This is a popular open-source library associated with NumFocus and developed by a group of FHE contributors. It offers support for major FHE schemes such as BGV, BFV, CKKS, TFHE, and FHEW, among others, and also enables multiparty operations.

[1] https://www.microsoft.com/en-us/research/project/microsoft-seal/.

[2] https://www.openfhe.org.

- **TFHE (C++)**[3]: The TFHE library provides an implementation of a very fast, gate-by-gate fully homomorphic encryption scheme. It allows for the evaluation of boolean circuits at impressive speeds, typically several orders of magnitude faster than other libraries.
- **Concrete**[4]: This library offers support for a custom-modified version of the TFHE encryption scheme.
- **PALISADE (C++)**[5]: The PALISADE library is unique for its breadth. It supports a wide variety of lattice cryptographic schemes, such as Learning with Errors (LWE), Ring-Learning with Errors (RLWE), and Number Theoretic Transform for Ring-based Unifying cryptography (NTRU), in addition to homomorphic encryption. The modular design of PALISADE allows researchers and developers to easily experiment with different schemes and parameters.
- **HElib (C++)**[6]: HElib provides a highly efficient implementation of the BGV scheme, with support for bootstrapping and the Gentry-Halevi-Smart optimizations [61], which substantially improve performance. It can handle both binary and approximate numbers and supports powerful operations like automatic noise management.
- **PySEAL (Python)**[7]: PySEAL wraps the functionality of the Microsoft SEAL library in a Pythonic interface, making it easier for Python developers to use. It allows users to perform arithmetic operations on encrypted data, using either the BFV or CKKS schemes.
- **Pyfhel (Python)**[8]: Pyfhel provides an even higher-level interface to homomorphic encryption, using the SEAL and HElib libraries as backends. It aims to make homomorphic encryption accessible to developers who are not familiar with C++ or the mathematics of homomorphic encryption.
- **FHEW (C++)**[9]: FHEW, like TFHE, allows for very fast evaluation of boolean circuits. It's designed to be easy to use, and its performance makes it suitable for many real-world applications of homomorphic encryption.
- **Lattigo (Go)**[10]: Lattigo is a Go library that supports the BFV and CKKS homomorphic encryption schemes. Its focus is on providing high performance, with many operations implemented in parallel to take advantage of multi-core processors.
- **HEaaN**[11]: It implements CKKS scheme and it can be used in several programming languages, for example, C++, Python, R.

[3] https://tfhe.github.io/tfhe/.
[4] https://github.com/zama-ai/concrete.
[5] https://palisade-crypto.org.
[6] https://github.com/homenc/HElib.
[7] https://github.com/Lab41/PySEAL.
[8] https://pyfhel.readthedocs.io/en/latest.
[9] https://github.com/lducas/FHEW.
[10] https://github.com/tuneinsight/lattigo.
[11] https://heaan.it/.

- $\Lambda \circ \lambda$ **(Haskell)** (pronounced "LOL")[12]: This library, developed in Haskell, offers support for FHE within the domain of ring-based lattice cryptography.
- **NFLlib (C++)**[13]: This library emerged from the European HEAT[14] project, an initiative aimed at investigating high-performance homomorphic encryption by harnessing low-level processor primitives.
- **cuHE**[15]: The CUDA Homomorphic Encryption Library (cuHE) is a library that leverages GPU (Graphics Processing Unit) acceleration for homomorphic encryption schemes and algorithms, all of which are defined over polynomial rings. This library offers remarkable performance coupled with a user-friendly interface, enhancing the efficiency of programmers. It encompasses both algebraic methods for homomorphic evaluation of circuits and well-optimized code suitable for single-GPU or multi-GPU machines (Table 3.1).

Table 3.1 List of the main active HE libraries and the implemented schemes

Library	Provider	BGV	BFV	FHEW	TFHE	CKKS	Bootstrap CKKS
HElib	IBM	X				X	
Microsoft SEAL	Microsoft		X			X	
PALISADE	Duality & a DARPA consortium	X	X	X	X	X	
HEAAN	Seoul National University					X	X
FHEW	Leo Ducas & Daniele Micciancio			X			
TFHE (Concrete library)	Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, Malika Izabachene, Zama				X		
FV-NFLlib	CryptoExperts		X				
Lattigo	EPFL-LDS		X			X	X

[12] https://github.com/cpeikert/Lol.
[13] https://github.com/quarkslab/NFLlib.
[14] https://heat-project.eu/.
[15] https://github.com/vernamlab/cuHE.

Some of these are no longer updated, but are worth mentioning to illustrate the efforts made in trying to make FHE widely used in practice. The range of available libraries for the implementation of FHE schemes proves the increasing interest and recognition of the significance of this field. The varied collection, spanning multiple programming languages and including several robust, high-performance options, makes available for developers and researchers significant tools to integrate and experiment with FHE in diverse applications.

These libraries, whether they implement ring-based lattice cryptography, provide GPU-accelerated computation, or support various FHE schemes, are important for advancing the state of homomorphic encryption, enabling its practical deployment, and broadening its impact on data privacy and security. These implementations also help to find the intricate trade-offs involved in FHE, including performance, complexity, and security, thereby paving the way for continued improvement and optimization.

The continued expansion and development of such libraries will undoubtedly employ further research and applications in this space, encouraging broader adoption of FHE and ultimately contributing to the realization of a more secure and privacy-preserving digital world. As the field continues to evolve and mature, these libraries and tools will remain at the forefront, serving as the foundation for the implementation of FHE schemes.

3.5 Advancements in Homomorphic Encryption

This section presents the recent advancements in homomorphic encryption, exploring the innovative techniques and methodologies that have been introduced to enhance its efficiency, security, and applicability. From novel encryption schemes to groundbreaking hardware optimizations, we will examine the strides made in this domain, shedding light on how these developments are shaping the future of secure data computation and privacy-preserving technologies.

3.5.1 Hardware Accelerators

HE represents an important topic nowadays with many benefits if adopted worldwide. Unfortunately, many HE schemes require high computational power, therefore, an important research direction is optimizing the HE schemes based on the hardware on which these run. Hardware optimization is essential in homomorphic encryption because it helps to address the significant performance challenges associated with performing computations on encrypted data:

- *Efficiency and Speed*: Homomorphic encryption schemes typically involve complex mathematical operations, such as modular arithmetic and polynomial evalua-

tions. These operations are computationally intensive and can be time-consuming, especially when dealing with large amounts of data. Hardware optimization techniques, such as specialized hardware accelerators or custom instruction sets, can significantly improve the efficiency and speed of these operations, making homomorphic encryption more practical for real-world applications.

- *Resource Utilization*: Homomorphic encryption requires substantial computational resources, including processing power, memory, and energy consumption. By optimizing the hardware, it is possible to utilize these resources more efficiently, reducing the overall computational burden. Dedicated hardware can be designed to perform specific operations required by the encryption scheme, taking advantage of parallel processing, custom algorithms, and optimized data paths. This can lead to improved performance and reduced resource requirements.
- *Scalability*: As the size of the encrypted data increases, the computational overhead also grows. Hardware optimization techniques can help in designing scalable solutions that can handle larger datasets without compromising performance. By leveraging hardware acceleration, it becomes possible to process more data within acceptable time frames, enabling the practical use of homomorphic encryption in scenarios where large-scale data processing is required.
- *Practicality*: Homomorphic encryption has the potential to revolutionize data privacy and security in various domains, such as cloud computing and data outsourcing. However, the computational overhead associated with homomorphic operations can be a significant barrier to its widespread adoption. By optimizing the hardware, the performance of homomorphic encryption can be improved, making it more practical and feasible for real-world applications.

Hardware optimization is necessary in homomorphic encryption, especially in Fully Homomorphic Encryption, due to the functional challenges that arise when used for complex computations such as machine learning. FHE enables computations over encrypted data without decryption, but it requires periodic noise reduction through bootstrapping, which is expensive and heavily bound by main memory bandwidth. Hardware optimization via parallel processors, such as AVX, GPU, FPGA, and ASIC, can significantly improve FHE's performance by enabling higher arithmetic intensity and lower memory bandwidth. For example, the OpenFHE[16] library is an open-source project that simplifies the complex cryptographic capabilities of FHE and supports different hardware acceleration technologies [62], creating opportunities for hardware acceleration providers to explore new markets. There are three main categories of hardware optimizations for FHE, depending of the hardware platform used: GPU (Graphics Processing Unit), FPGA (Field Programmable Gate Array), and ASIC (Application Specific Integrated Circuit).

One of the preliminar works is Doróz et al. [63], which eficientize Gentry's scheme [5] and its optimized version [21]. The proposed design showcases an enhanced multi-million bit multiplier that leverages the Schonhage Strassen multiplication algorithm. The architecture also incorporates a variety of optimizations, such as spectral techniques and a precomputation strategy, to substantially enhance the performance of

[16] https://www.openfhe.org.

the entire system. When implemented using 90 nm technology, the architecture successfully performs encryption, decryption, and recryption operations in 18.1, 16.1 milliseconds, and 3.1 s, respectively. Furthermore, the design requires <30 million gates, demonstrating its compact footprint.

A work in this direction is Yang et al. [64] that targets the Paillier cryptosystem applied with machine learning algorithms. In this investigation, the authors develop and build an Field-Programmable Gate Array (FPGA) circuit utilizing high-level synthesis (HLS) and a high-level programming language for flexibility. This method makes the algorithm and operations parametric and portable. The authors also want to create an analytical model that can predict encryption performance and optimize it across various dimensions. Because modular multiplication (ModMult) accounts for the vast majority of processing in the Paillier cryptosystem, the authors' primary emphasis is on developing a compact architecture for this operation. The Montgomery method is used by the authors, which is FPGA-friendly since it avoids the requirement for integer division operations. To get the greatest throughput, the authors identify the essential elements that impact overall encryption/decryption throughput on an FPGA device and perform detailed optimization of Paillier processors in terms of clock cycle, resource use, clock frequency, and memory utilization. The hardware modules developed by the authors are built as OpenCL kernels and incorporated into FATE as an encryption library. To reduce kernel invocation overhead, each kernel conducts encryption/decryption for a batch of data, and kernels are queued in the OpenCL command queue to overlap data transmission with computation and conceal delay. While ensuring security and accuracy, the authors' suggested encryption system is universal and does not need any modifications to the model. The authors undertake thorough tests on their proposed framework, demonstrating that it decreases iteration time for training linear models by up to 26% and encryption time by 71% in each iteration. When compared to software methods, the authors' hardware framework gives an acceleration ratio of 10.6 for encryption and 2.8 for decryption. Their ModMult circuit outperforms previous FPGA solutions in terms of DSP efficiency, with similar execution latency but reduced DSP block use.

A work that also uses FPGA is Su et al. [65], which proposed a hardware version of the BGV tiered FHE method that is efficient, being the first comprehensive FPGA-based RLWE accelerator for the BGV algorithm [65]. Unlike efforts priors to this work that relied on various FHE techniques, their architecture allows both homomorphic encryption and homomorphic evaluation processing, allowing to tune it to particular application needs. The authors used multi-layer parallelism to speed up processes, beginning with the circuit level and advancing to the arithmetic block level. The proposed architecture provides a big increase in the performance of polynomial multiplication over the ring. Based on the Number Theoretic Transform (NTT), the Negative Wrapped Convolution (NWC) method is created, which includes a four-level pipeline and a single-round iterative structure. This optimized structure achieves an excellent mix of performance and space usage and provides a resource-efficient and high-performance modular reduction technique that requires half the resources that the Barrett reduction method does. The paper also discusses the hardware design of the KeySwitch and ModSwitch modules, which are required for

implementing the leveled BGV FHE method. Moreover, the authors present the first hardware structure for the ModSwitch module, which employs a reduced modulus to decrease noise in the ciphertext of homomorphic evaluation.

Another important recent work in this direction is Geelen et al. [66], a hardware accelerator for the BGV scheme. Here, the authors propose BASALISC, which is a RISC (Reduced Instruction Set Architecture) architecture with three abstraction layers. It is especially intended for use in a 1 GHz Application-Specific Integrated Circuit (ASIC) and being taped out on a 150 mm^2 die utilizing a 12 nm GF process. BASALISC's four-layer memory architecture, which includes a conflict-free inner memory layer, is a standout feature. This inner memory layer allows for 32 Tb/s radix-256 NTT calculations without pipeline pauses. BASALISC also has conflict-resolution permutation hardware that may be utilized for BGV automorphisms without losing performance. A special multiply-accumulate unit is also included in the design to speed up BGV key switching. BASALISC's compute units and inner memory layers are both built using asynchronous logic, enabling them to run at varying rates to optimize each function. A proprietary compiler with a performance and accuracy simulator comprise the BASALISC toolchain. Various studies are performed to evaluate BASALISC, including examining its physical realizability, mimicking and formally proving its fundamental functional units, and testing its performance against a set of benchmarks. BASALISC is evaluated in one assessment for a single iteration of logistic regression training on encrypted data. This application converts to 513 bootstraps, 900 K high-level BASALISC instructions, or 27B low-level BASALISC instructions. According to the statistics, BASALISC is only 3,500 times slower than an Intel Xeon class Central Processing Unit (CPU) operating without data encryption. Furthermore, in an individual bootstrapping operation, BASALISC outperforms the widely used software FHE library, HElib. The obtained speedup is 4,000 times quicker than HElib. Overall, BASALISC demonstrates its potential by providing efficient performance and increased speed in a variety of computing activities, making it a suitable architecture for homomorphic encryption applications [66].

Other works related to hardware optimizations for HE are [67–69], etc.

It is worth mentioning that FHE has gained the atention to different national agencies that fund projects related to it. For example, The Defense Advanced Research Projects Agency (DARPA) Data Protection in Virtual Environments (DPRIVE) program is focused on developing privacy-enhancing technologies, including FHE. Under this program, DARPA has awarded contracts to various companies to design and develop hardware accelerators for FHE computations. One of the companies, Duality Technologies, has been awarded a $14.5m contract to develop an ASIC named *TREBUCHET* for FHE computations[17] With TREBUCHET, the major FHE schemes are accelerated at BGV, BFV, CKKS, FHEW, etc.) at at least 128-bit security [70]. Intel and Microsoft is also working on an ASIC accelerator for FHE under

[17] Duality Technologies Awarded $14.5M DARPA Contract to Develop World's Fastest Privacy-Preserving Hardware Accelerator, https://www.prnewswire.com/il/news-releases/duality-technologies-awarded-14-5m-darpa-contract-to-develop-worlds-fastest-privacy-preserving-hardware-accelerator-301221040.html.

this program[18], [19]. The program will span several phases, starting with the design, development, and verification of foundational IP blocks.

3.5.2 *Quantum Homomoprhic Encryption*

Homomorphic encryption, as it is traditionally defined, is not inherently resistant to attacks from quantum computers. Quantum computers have the potential to break many of the asymmetric cryptographic algorithms currently in use, such as RSA and elliptic curve cryptography (ECC), which are fundamental components of homomorphic encryption schemes. Quantum computers can exploit Shor's algorithm [71] to efficiently factor large numbers or solve the discrete logarithm problem, rendering these cryptographic schemes insecure.

However, there is ongoing research into developing quantum-safe or post-quantum homomorphic encryption schemes that are resistant to attacks from quantum computers. These schemes aim to provide security even in the presence of a powerful quantum adversary. Various cryptographic primitives, such as lattice-based cryptography, code-based cryptography, multivariate cryptography, and others, are being explored as potential building blocks for post-quantum homomorphic encryption. Researchers and cryptographic experts are actively investigating the design and implementation of quantum-resistant homomorphic encryption schemes to ensure secure computation in a post-quantum computing era. The need for research in this direction is also sustained by the NIST Post-Quantum Cryptography standard which is user development (see Sect. 3.4.2.2 for more details).

To be quantum-resistant, an encryption scheme should possess certain properties that make it secure against attacks from quantum computers. The crucial aspect is that quantum-resistant encryption schemes should be designed to resist against attacks from quantum algorithms, such as Shor's algorithm, which can efficiently factor large numbers or solve the discrete logarithm problem.

There are several types of homomorphic encryption schemes that are considered being quantum resistant:

- *Lattice-Based Homomorphic Encryption*: Lattice-based cryptography is a leading candidate for post-quantum security. Various lattice-based HE schemes, such as the BGV [14] scheme and its variants, have been proposed. These schemes rely on the hardness of lattice problems, which are considered to be resistant to quantum algorithms like Shor's algorithm.

[18] Intel And Microsoft Partner For DARPA DPRIVE 'Holy Grail' Cybersecurity Project For Homomorphic Encryption, https://hothardware.com/news/darpa-partners-with-intel-and-microsoft-for-cybersecurity-project.

[19] Intel Completes DARPA DPRIVE Phase One Milestone for a Fully Homomorphic Encryption Platform, https://community.intel.com/t5/Blogs/Products-and-Solutions/HPC/Intel-Completes-DARPA-DPRIVE-Phase-One-Milestone-for-a-Fully/post/1411021.

- *Code-Based Homomorphic Encryption*: Code-based cryptography, specifically code-based encryption (CBE), is another promising approach for quantum-resistant encryption. Code-based HE schemes, such as the NTRUEncrypt scheme [49], utilize error-correcting codes and other mathematical structures to provide encryption and homomorphic capabilities.
- *Multivariate Homomorphic Encryption*: Multivariate cryptography involves mathematical problems that are believed to be resistant to quantum attacks. Homomorphic encryption schemes based on multivariate cryptography, such as the Hidden Field Equations (HFE) scheme and its variants [72], have been explored for post-quantum resistance.
- *Supersingular Isogeny-Based Homomorphic Encryption*: Supersingular isogeny-based cryptography is an emerging field that offers potential for post-quantum security. Recent research has shown progress in developing homomorphic encryption schemes based on isogenies, such as the SIDH-based (Singular Isogeny-based Diffie-Hellman) HE scheme [73], which rely on the hardness of isogeny problems.
- *Hash-Based Homomorphic Encryption*: Hash-based cryptography, which relies on the properties of cryptographic hash functions, is another area of research for post-quantum encryption. Hash-based HE schemes, such as the WOTS+ based scheme [74], leverage hash functions to provide homomorphic properties.

Before going into e deeper view of homomorphic schemes that can be quantum-resistat, firstly, we will see what quantum computing actually is. The information within this section related to quantum computing is from Kaye et al. [75], LaPierre [76]. Quantum computing is a field of computing that leverages the principles of quantum mechanics to perform computational tasks. Unlike classical computers, which use binary bits (0 s and 1 s) as the fundamental unit of information, quantum computers use quantum bits, or qubits, which can represent 0, 1, or a superposition of both states simultaneously. Quantum computing harnesses the unique properties of qubits to perform certain computations more efficiently than classical computers. At a low level, quantum computing is based on the following components [75, 76]:

- *Qubits*: Qubits are the basic building blocks of quantum computers. They can exist in a superposition of states, representing both 0 and 1 simultaneously. This superposition allows for parallel processing and can significantly increase computational power.
- *Quantum Gates*: Quantum gates are analogous to the logical gates used in classical computing. These gates manipulate the qubits, allowing for operations such as superposition, entanglement, and measurement. Quantum gates are used to perform computations and transform the quantum state of the qubits.
- *Entanglement*: Entanglement is a unique property of quantum mechanics that allows multiple qubits to become correlated in such a way that the state of one qubit is directly linked to the state of another, regardless of the physical distance between them. Entanglement is a crucial resource in quantum computing and enables certain computational advantages.
- *Quantum Algorithms*: Quantum algorithms are designed to leverage the properties of qubits, such as superposition and entanglement, to solve specific computational

problems more efficiently than classical algorithms. Prominent examples include Shor's algorithm for integer factorization and Grover's algorithm for unstructured search.

- *Quantum Error Correction*: Quantum computers are susceptible to errors due to noise and interference from the environment. Quantum error correction techniques are used to detect and correct errors that occur during quantum computations, ensuring the reliability and accuracy of the results.

A qubit, or quantum bit, is the fundamental unit of quantum information. It is the quantum analog of a classical bit. While a classical bit can be in one of two states (0 or 1), a qubit can be in a superposition of states. The state of a qubit can be represented as Kaye et al. [75], LaPierre [76]:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{3.16}$$

Here, $|0\rangle$ and $|1\rangle$ are the basis states, and α and β are complex numbers. The coefficients α and β are probability amplitudes and can be used to calculate the probability of the qubit collapsing to either the $|0\rangle$ and $|1\rangle$ state when measured. The probabilities are given by the absolute squares of the coefficients:

$$\begin{aligned} P(0) &= |\alpha|^2 \\ P(1) &= |\beta|^2 \end{aligned} \tag{3.17}$$

The sum of the probabilities must equal 1, which leads to the normalization condition:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{3.18}$$

This condition ensures that the qubit will collapse to one of the basis states upon measurement.

Qubits can also be entangled, a unique quantum property that allows particles to have a much higher correlation than is possible classically. If two qubits are entangled, the state of one qubit is directly related to the state of the other, no matter how far apart they are. Quantum entanglement is a unique phenomenon that comes from quantum mechanics where two or more particles become linked and instantaneously affect each other's state no matter how far apart they are. This correlation is stronger than what can be explained by classical physics.

Entangled particles are described by a single wave function. If we consider two qubits, they can be entangled to form a state such as:

$$|\psi\rangle = 1/\sqrt{2}(|00\rangle + |11\rangle) \tag{3.19}$$

The Eq. (3.19) is known as a Bell state, one of the simplest examples of entangled states [75, 76]. Here, $|00\rangle$ and $|11\rangle$ are the basis states. The coefficients $1/\sqrt{2}$ are probability amplitudes. If one of the qubits is measured and found in state $|0\rangle$, the

other qubit will also be in state $|0\rangle$, even if they are light-years apart. The same applies for the state $|0\rangle$. It's important to note that entangled states are a superposition of states, and the specific outcome isn't determined until the measurement. However, once one of the entangled particles is measured, the state of the other particles is instantly determined, no matter the distance between them.

Quantum gates are the basic operations in quantum computing that act on qubits to change their states. Unlike classical gates, which are deterministic and perform operations like AND, OR, and NOT, quantum gates are represented by unitary matrices and can perform operations that alter the phase and probability amplitudes of qubit states. Examples of common quantum gates [75, 76]:

- **Pauli-X Gate (Bit-Flip Gate)**: This gate is the quantum equivalent of the classical NOT gate. It flips the state of a qubit from $|0\rangle$ to $|1\rangle$ and vice versa. It is represented by the Pauli-X matrix:
$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$
- **Pauli-Y Gate**: This gate applies both a bit-flip and a phase-flip to a qubit. It is represented by the Pauli-Y matrix:
$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$
- **Pauli-Z Gate (Phase-Flip Gate)**: This gate flips the phase of a qubit. It is represented by the Pauli-Z matrix:
$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$
- **Hadamard Gate**: This gate creates superposition. It maps the basis state $|0\rangle$ to $(|0\rangle + |1\rangle)/\sqrt{2}$ and $|1\rangle$ to $(|0\rangle - |1\rangle)/\sqrt{2}$. It is represented by the Hadamard matrix:
$$H = 1/\sqrt{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$
- **CNOT Gate (Controlled NOT Gate)**: This is a two-qubit gate that performs a NOT operation on the second qubit (target) if the first qubit (control) is in the state—1. It is represented by the matrix:
$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

The action of a quantum gate on a qubit is represented by multiplying the state vector of the qubit with the matrix of the gate. The resulting state vector represents the state of the qubit after the operation.

One of the early works related to quantum homomorphic encryption is Broadbent and Jeffery [77]. Here, a public key quantum homomorphic encryption scheme is defined as having the following algorithms:

- $\texttt{QHE.KeyGeneration}(1^K) \rightarrow (pk, sk, \rho_{evk})$: based on the security parameter $k \in \mathbb{N}$ the public key pk ans private key sk are generated. These are regular encryption keys, but are used for encryption and decryption of quantum information. The third key, ρ_{evk} is the evaluation key, which is in a quantum state. Its state is consumed after it is used to evaluate quantum circuits over encrypted data.
- $\texttt{QHE.Encryption}(\rho, pk) \rightarrow \sigma$: this algorithm encrypts the quantum plaintext ρ using the public key. It results in the ciphertext σ.
- $\texttt{QHE.Evaluation}(\sigma, \rho_{evk}, QC) \rightarrow \sigma'$: this algorithm evaluates the quantum circuit QC applied on the quantum ecnrypted text σ, using the quantum evaluation key ρ_{evk}. The result is the quantum ciphertext σ'.
- $\texttt{QHE.Decryption}(\sigma', sk) \rightarrow \rho$: this algorithm decrypts the evaluated quantum ciphertext σ' using the secret key sk. The output is the quantum plaintext ρ.

Also, the form of a homomorphism is changed for quantum computations, as follows [77].

Definition 3.8 Let ρ be a quantum plaintext. If for any quantum circuit QC, and the Θ quantum channel of the circuit Qc there is a negligible function η for which the property from below is satisfied, then the QHE has homomorphic properties.

$$Pr[\texttt{QHE.Decryption}(\texttt{QHE.Evaluation}(\texttt{QHE.Encryption}(\rho, pk), \rho_{evk}, QC), sk) \neq \Theta(QC, \rho)] \leq \eta(k) \tag{3.20}$$

In Zhang et al. [78] the authors propose a quantum MPC protocol that has homomorphic properties, which compares information. Participants involved in this protocol may compare encrypted information bits. The responsibility of comparing encrypted data is assigned to the Third Party (TP), a computationally capable yet virtually dishonest server. The trusted key center (TKC), is in charge of distributing and upgrading keys, decrypting assessed data, and accurately publishing comparison findings. Assume there are n participants, indicated by $P_i, i = 1, 2, \ldots, n$, who want to see whether their m pieces of private information $M_{(}ij)(i = 1, 2, \ldots, n, j = 1, 2, \ldots, m)$ are equal. They do not, however, wish to reveal the contents of their private information. To overcome this, the authors propose a QHE technique that enables the server to compute on encrypted data [78]. After decryption, the estimated result obtained on the original data is as requested and may be honestly stated to all parties. Furthermore, Measurement Device Independent Quantum Key Distribution (MDI-QKD) may be used to secure the key distribution process. Private information is encrypted using the Quantum One-Time Pad (QOTP) in the context of quantum data. As the pad's key for a quantum state ρ, and $a, b \in \{0, 1\}$ are picked at random. The quantum information state is encrypted using Pauli operations, resulting in the ciphertext σ, a totally mixed state. Each quantum state encryption creates a new random Pauli key. Only those who have the decryption key may decipher the ciphertext

and get access to significant information. As a result, intercepting the whole ciphertext would be pointless. The data is hidden inside a totally mixed state, assuring its security. The encrypted quantum information is achieved using the formula:

$$\sigma = \sum_{a,b\in\{0,1\}} \frac{1}{4} X^a Z^b \rho (X^a Z^b) = \frac{I_2}{2}.$$

The protocol works as follows (we keep the notations as in the original paper) [78]:

- `KeyGeneration`: the entities involved in the system are the participants P_i and TKC. The participants are using the trusted preparation process in order to randomly generate one of the quantum states $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$. Further, the prepared quantum states are sent to the TP, which conducts a Bell state measurement for every pair of quantum states $|\varphi_{p_i}\rangle|\varphi_{p_i}\rangle$. Then the results $|\phi_{p_i}{}^{\pm}\rangle|\phi_{p_i}{}^{\pm}\rangle$ are sent to the participants. The basis information related to the prepared quantum states is published by each P_i and TKC for the authenticated channel, which is in classical form. From the measurements made by TP, only the qubits resulting in the same preparation basis are kept. Therefore, the key at this point is the shifted key, and from it, a part is announced. Then, after post-processing, the value of the error rate is verified. If its value is used the threshold it means that there is no eavesdropped on the channel, therefore, the communication is secure. The encoding rules used by the all parties involved in this algorithm are $|0\rangle, |+\rangle \to 0$; $|1\rangle, |-\rangle \to 1$. The security key obtained by all P_i and TKC is denoted with $ek_i = (a_i, b_i), a_i, b_i \in \{0, 1\}$.
- `Encryption`: each participant P_i uses ek_i as a one-time encryption key to encrypt the first bit of the private value that will be compared with the others. The encrypted value is then sent to TP, which as a final result has:

$$(X^{a_1} Z^{b_1} \otimes \ldots (X^{a_n} Z^{b_n}) |\phi_1\rangle \ldots |\phi_n\rangle$$

- `HomomorphicEvaluation`: the encrypted states are owned by any two participants, let them be $P_l, l = 1, \ldots, n$ and $P_k, k = 1, \ldots, n$, which are evaluated by TP. The control bit used in a *CNOT* operation is the particle of P_l, while the target bit is the particle of P_k. The result of the *CNOT* operation applied to inputs is then sent to TKC.

$$CNOT(X^{a_k} Z^{b_k} \otimes X^{a_l} Z^{b_l}) |\phi_k\rangle |\phi_l\rangle = (X^{a_k} Z^{b_k \oplus a_l} \otimes X^{a_k \oplus a_l} Z^{b_l}) |\phi_k\rangle |\phi_k \oplus \phi_l\rangle.$$

When the number of participants $n \geq 3$ is reached, an auxiliary particle $|0\rangle$ is introduced, and TP executes a series of actions dependent on the sequence in which the encrypted states are sent. To begin, the first participant's encrypted state is used as the control bit, and the encrypted state of the second participant to be compared is used as the target bit to apply a CNOT gate. This target bit is then utilized as the control bit, and the auxiliary particle $|0\rangle$ is used as the target bit for the CNOT gate. Following that, the first participant's encrypted state is used as the

control bit once again, and the encrypted state of the third person to be compared is used as the target bit to repeat the aforementioned action. This method is repeated until all participants' encrypted states, save the first, have been utilized as target bits and control bits to conduct the matching CNOT operation. When the comparison is finished, the results are provided to TKC.

- `Decryption`: for any two participants, the encryption key is updated by *PKC* in order to obtain the decryption key $dk_i = (a'_k, b'_k, a'_l, b'_l)$, as follows: $ek_k = (a_k, b_k)$, $ek_l = (a_l, b_l) \rightarrow$ [update]$dk_i = (a'_k, b'_k, a'_l, b'_l) = (a_k, b_k \oplus b_l, a_k \oplus a_l, b_l)$. Further, dk_i is used to decrypt the evaluated encrypted quantum state. Lastly, the auxiliary particle is verified: id its value is $|0\rangle$, the secret information is the same, otherwise, the secret information is different. When there are $n \geq 3$ participants, all particles, including the auxiliary particle, are measured in the basis $\{|0\rangle, |1\rangle\}$, with the exception of the particle owned by the first participant who consistently holds the control position. The result R_j, $j = 1, 2, \ldots, m$ is the sum of the n measurement results c_i, $i = 0, 1, \ldots, n-1$). If $R_j \neq 0$ is returned, TKC states that the secret bit possessed by participants is different, and the process is completed. If not, the protocol compares the next bit of private data from multiple participants until the m-th bit is compared. If the sum of $\sum_{j=1}^{m} R_j = 0$, then it is declared that these participants' private data is identical.

The authors prove that their proposed quantum MPC protocol is secure against participant's attack, TP's attack, outside attack. The protocol reduces not just the utilization of quantum resources but also the assumed security requirements for the third-party server. Along with a major improvement in the protocol's security, the IBM Q experimental platform has proven its effectiveness. The statistical results support the protocol's efficacy.

The scheme that includes quantum homomorphic encryption [78] and presented here is one of the many works in this area. To work on qubit operations and quantum circuits, quantum homomorphic encryption needs certain considerations. The intrinsic properties of quantum technology increase the noise problems associated with encryption. There are current ways that allow for the limited use of this kind of encryption. In Tan et al. [79], for example, the authors developed a quantum homomorphic encryption approach that restricts the information available to an unauthorized observer. In Broadbent and Jeffery [80] have also presented a tried-and-true homomorphic encryption approach for circuits with low T-gate complexity. This approach uses a quantum one-time pad [81] as the basis for encryption over Clifford group circuits, and it becomes completely homomorphic by adding an encryption mechanism for T-gates.

However, according to Yu et al. [82] any Quantum Fully Homomorphic Encryption (QFHE) method with complete security would need exponential storage space if interactions were not allowed. To put it another way, it is impossible to build an efficient QFHE system with complete security. As a result, some researchers have focused on QFHE schemes under more relaxed conditions, such as scenarios in which interaction is allowed [79, 83], the security level can be reduced to computational security [84], or the client provides enough copies of encrypted ancillary states for

the servers [85]. However, supplying too many copies may result in information leaks, preventing complete security from being achieved. In Liang [83], the authors presented another QFHE technique based on Universal Quantum Computing (UQC). This technique takes use of the advantages of UQC in that the decryption key varies from the encryption key, and both are kept secret, along with the shared key. UQC essentially assures the scheme's security. Interactions, on the other hand, are essential, and their number is equal to the number of T-gates in UQC.

3.5.3 Multi-party Computations

Multi-party computation (MPC), also known as secure multi-party computation, is a cryptographic technique that enables a group of parties to jointly compute a function over their private inputs while preserving the privacy of individual inputs. It allows multiple participants to collaboratively perform computations without revealing their private data to each other.

In MPC, a group of participants, each holding their private input, want to compute a function collectively without disclosing their individual inputs. The key concept in MPC is to distribute the computation among the parties in a way that allows them to perform operations on their private inputs without exposing them to others. More specifically, MPC involves a set of parties $(P_1, P_2, \ldots, P_n)$ who each hold their private input values $(x_1, x_2, \ldots, x_n)$. The goal is to compute a desired function $f(x_1, x_2, \ldots, x_n)$ without exposing any individual's input to the other parties [86].

In general, an MPC protocol involves several steps [86]:

- Input Sharing: Each party privately shares their input using a secret sharing scheme. This involves splitting their input into multiple shares, distributed among the parties. The secret sharing ensures that no single party has access to the complete input.
- Secure Computation: The parties collaboratively execute a secure computation protocol to compute the desired function on the shared inputs. The protocol allows the parties to perform computations while preserving the privacy of individual inputs. It typically involves cryptographic techniques like homomorphic encryption, garbled circuits, or secure function evaluation.
- Output Reconstruction: Once the computation is completed, the parties jointly reconstruct the output of the function without revealing their individual inputs. This is done through a protocol that combines the output shares held by the parties to obtain the final result.

MPC protocols have several challenges, such as communication overhead, computation complexity, and security concerns. One of the primary challenges is the issue of privacy. In an MPC protocol, multiple parties collaborate to compute a function over their inputs while keeping those inputs private. However, ensuring privacy in the presence of malicious parties who may collude or deviate from the protocol

is a significant challenge. Another challenge is the issue of efficiency. MPC protocols often require complex mathematical operations and communication between parties, which can be computationally intensive and slow, especially for large-scale computations.

One way to address these challenges is through HE, which can be used in MPC protocols to reduce communication overhead and computation complexity. HE allows parties to perform computations on encrypted data, thus reducing the amount of data that needs to be communicated between parties. This reduces communication overhead and computation time, making MPC protocols more efficient. HE can also improve security in MPC protocols by reducing the number of times parties need to decrypt their data. Decrypting data increases the risk of data leakage, which can compromise the security of the protocol. With HE, parties can perform computations on encrypted data without decrypting it, thus reducing the risk of data leakage.

Additionally, HE can be used to perform operations that are not possible in traditional MPC protocols, such as multiplication of encrypted data. This enables more complex computations to be performed on encrypted data, further increasing privacy and security.

As we have seen, HE can be used in the step of secure computation of the MPC protocol, Therefore, with HE, each party can encrypt their private input using a suitable HE scheme. The encrypted inputs can then be securely shared among the parties. By including the homomorphic properties of the encryption scheme, the parties can perform computations on the encrypted data without decrypting it. The result is an encrypted output that, when decrypted, yields the correct result of the computation without revealing the individual inputs.

There are several HE schemes that have been used in MPC protocols:

- Paillier Encryption: It has been used in MPC protocols to perform computations on encrypted data while maintaining privacy. Paillier encryption is well-suited for additive computations, such as summing encrypted values or performing secure averaging.
- ElGamal Encryption: ElGamal encryption is another PHE scheme widely used in MPC, being often combined with additional techniques, such as secret sharing or garbled circuits, to achieve secure MPC.
- BGV scheme: it has been utilized in MPC protocols to enable general-purpose secure computations. The BGV scheme provides a higher level of expressiveness compared to PHE schemes but typically involves higher computational overhead.
- BFV scheme: being a balance between efficiency and functionality, it enables computations on encrypted data and offers better performance compared to fully homomorphic encryption schemes.
- GSW scheme, has also been employed in MPC settings. Lattice-based encryption provides post-quantum security and can be used to perform computations on encrypted data while preserving privacy.

The notion of reducing interaction in MPC via the use of threshold homomorphic encryption has its origins in the work of Franklin and Haber [87], which was

further improved by in Cramer et al. [88]. Recently, certain examples of multi-party additive-homomorphic encryption have proved beneficial in aiding distributed machine learning use-cases [89, 90], highlighting the potential significance of a general and useable FHE solution. This is the motivating force behind the works [91, 92], in which are suggested several multi-party schemes in which the secret key is additively shared among parties and investigated the theoretical MPC solutions permitted by these schemes. Despite their importance, these efforts have not received as much application as method (a). This might be owing to a lack of efficient implementations of Learning with Errors [93] (LWE)-based homomorphic schemes, on which these methods were built.

3.5.4 Zero-Knowledge Proof

Zero-knowledge proof (ZKP) is a cryptographic protocol that allows multiple parties to collectively prove the validity of a statement without revealing any additional information beyond the truth of the statement. In a multi-party ZKP, multiple participants work together to prove a common statement while preserving privacy and maintaining the confidentiality of their individual inputs. In general, a multi-party ZKP protocol involves the following steps [94, 95]:

- Statement Setup: The parties collectively agree on a statement that they want to prove, such as the validity of a mathematical computation or the satisfaction of certain conditions.
- Shared Inputs: Each party holds a private input related to the statement. These inputs may be secret values or information that needs to remain confidential.
- Interactive Proof Generation: The parties engage in an interactive protocol to generate a zero-knowledge proof of the statement's validity. They exchange messages and computations to convince a verifier that the statement is true without revealing any additional information about their private inputs.
- Verification: The proof generated by the parties is verified by a separate entity, known as the verifier, who evaluates the proof to determine its validity. The verifier can confirm the truthfulness of the statement without learning the specific details of each party's input.

With these steps, the ZKP protocols have the following characteristics [96]:

- Completeness: If the statement is true, all honest parties can generate a valid proof that will be accepted by the verifier.
- Soundness: If the statement is false, no group of dishonest parties can produce a convincing proof that will be accepted by the verifier with a high probability.
- Zero-Knowledge: The proof leaks no information about the private inputs of the parties, except for the validity of the statement being proved.

Multi-party ZKP protocols are important in cryptography because provide a means for parties to collaborate and prove statements collectively, while preserving privacy

and confidentiality. They find applications in scenarios such as secure multiparty computations, where multiple participants need to jointly prove the correctness of their computations without revealing the underlying data or inputs.

While HE and ZKPs have different purposes, they can be used together in certain scenarios to achieve additional privacy and security goals. For example, ZKPs can be employed within an HE-based secure computation protocol to prove specific properties or assertions about the encrypted data without revealing the actual data. HE can be used in Verifiable Computation with Fully Homomorphic Encryption (VCFHE), which is a framework that combines FHE and zero-knowledge proofs to achieve verifiable computations [97]. It allows a prover to compute a function on encrypted data and generate a zero-knowledge proof of the correctness of the computation without revealing the underlying data or intermediate values. This combination of FHE and ZKPs enables privacy-preserving computations while providing verifiability.

3.6 Case Studies and Practical Applications

In this section, we investigate practical applications of HE, to discover its potential and impact in different domains. While the theoretical underpinnings of HE can be complex, its application and large-scale use can transform the field of data privacy and security. Despite the computational complexities associated with HE, it offers compelling solutions in scenarios where data confidentiality is crucial, such as in healthcare, finance, cloud computing, and more. Through a series of case studies, we will explore how this groundbreaking technology is already being incorporated into real systems, illustrating its potential to unlock new capabilities while preserving data privacy. These real-world applications prove the importance of HE, offering a perspective into a future where secure computation on encrypted data is not just possible, but a standard practice.

3.6.1 Cloud Computing and Big Data

Homomorphic encryption can address the security concerns related to cloud computing and presented in Sect. 2.3.1 by ensuring that data remains encrypted while being processed or analyzed, therefore preserving the confidentiality of the information. Homomorphic encryption can also be used in scenarios involving the outsourcing of computational tasks or facilitating the secure outsourcing of big data processing tasks to the cloud or other third parties. By encrypting the data prior to transmission for processing, computations can be performed on the encrypted data itself. Moreover, many industries are subject to stringent regulations mandating the encryption of certain types of data. Homomorphic encryption would help organizations to leverage cloud services and big data analytics without contravening these regulations. In multi-party computation scenarios, where data from multiple sources needs to

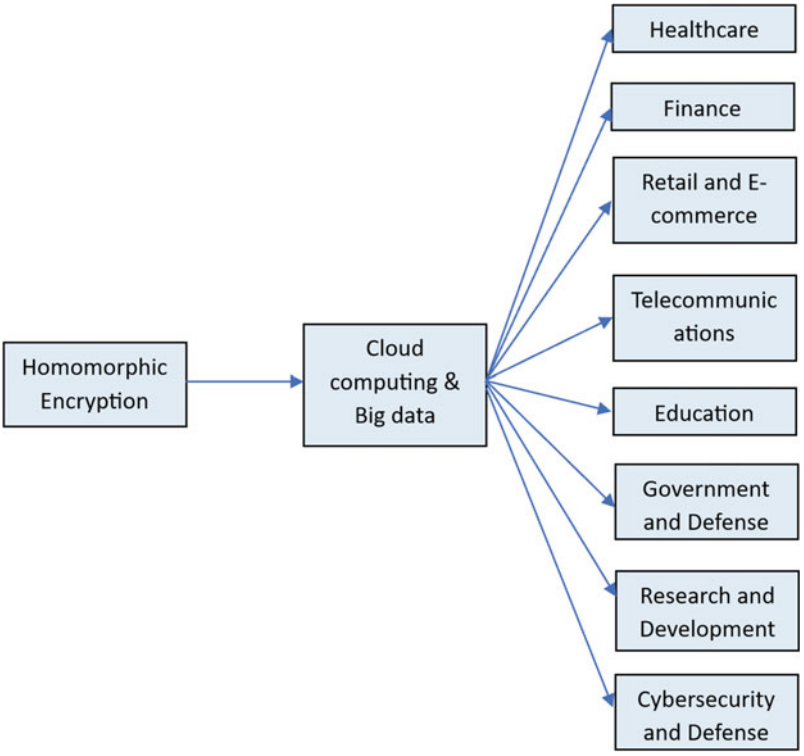

Fig. 3.2 Main domains and industries from cloud computing where HE can be included

be aggregated and computed, each party typically wishes to maintain the privacy of their individual input and, by using HE, the computation of functions across multiple parties is made without the need for any party to reveal their inputs to the others. Finally, homomorphic encryption facilitates the secure sharing of encrypted data with third parties for analysis or monetization, without relinquishing control of the raw data. This ensures that data can be utilized without compromising privacy or security. Figure 3.2 shows the main domains and industries from cloud computing where HE can be included.

Homomorphic encryption can have an important role in cloud computing and big data by addressing the privacy and security concerns associated with data processing and storage, as follows:

- Data Confidentiality: Homomorphic encryption enables computations to be performed on encrypted data without the need for decryption. This ensures that sensitive information remains confidential throughout the entire data lifecycle [98, 99].
- Secure Collaborations: With homomorphic encryption, multiple parties can collaborate and perform computations on encrypted data without exposing the under-

lying information. This facilitates secure collaborations in cloud computing and big data scenarios [99].

- Outsourcing Computations: Homomorphic encryption allows for the outsourcing of computations to cloud service providers while preserving data privacy. The encryption ensures that the data remains protected even when processed by third-party servers [99].
- Privacy-Preserving Machine Learning: Homomorphic encryption can be applied in machine learning algorithms to perform computations on encrypted data without compromising data privacy. This enables secure data analysis and protects the confidentiality of sensitive information [98].
- Potential Applications in Nuclear Safeguards: Homomorphic encryption has the potential to be applied in nuclear safeguards, where data confidentiality is of utmost importance. It allows for secure computations on sensitive nuclear data without exposing the details to unauthorized individuals [99].

An example of a cloud computing framework that includes homomorphic encryption is presented in Ibtihal et al. [100], designed for image protection. The contributions of the work are as follows: (i) An OpenStack-based private cloud was implemented with a focus on providing encryption services and validating the concept of Encryption as a Service. (ii) A custom program was developed in the C programming language to facilitate the encryption and decryption of images using the Paillier cryptosystem. The program was implemented on the Nova hypervisor for efficient execution. (iii) Another program, also developed in C, was created to demonstrate the homomorphic property of the encryption scheme. This program utilized the discrete wavelet transform (DWT) implemented in the encrypted domain to evaluate the effectiveness of the scheme. The advantage of the proposed system lays in the fact that it is performant and can be used in real applications

3.6.2 Machine Learning and Artificial Intelligence

In this context of AI and ML presented in Sect. 2.3.2, homomorphic encryption is important because mainly it enables computations on encrypted data, thereby preserving data privacy. In the era of data-driven decision-making, ML and AI models often require access to sensitive data, so homomorphic encryption allows this data to remain encrypted while being used for training or inference, thus privacy and confidentiality are ensured. Important applicability of HE is in federated learning, a decentralized approach to ML where the model is trained across multiple devices or servers holding local data samples. In this case, homomorphic encryption would ensure that model updates are securely shared between devices or servers, without revealing sensitive information. Moreover, homomorphic encryption facilitates secure outsourcing of computation. Given the fact that ML and AI require power-consuming computations, these tasks can be securely outsourced to more powerful machines or cloud-based systems without exposing raw data.

When constructing secure machine learning models, there should be taken into consideration the security of the input data, output data, and the model itself. Several ML/AI models included encryption schemes for data privacy that have homomorphic properties.

A study in this direction is Song and Huang [101]. According to the authors, this work uses homomorphic encryption technology to secure user privacy (applied to images owned by the users) at the inference stage, which is critical for privacy protection, with the goal of addressing the problem of privacy leakage when users interact with cloud providers, based on the CKKS scheme [43]. The cloud is unable to access any user information throughout the inference process, protecting user privacy. While there are existing schemes that combine homomorphic encryption with a neural network inference stage, the authors identified potential improvements in these schemes' integration of the network structure with the ciphertext structure, which could effectively reduce the complexity of ciphertext transmission across different model layers, therefore, they suggest a more efficient ciphertext allocation for the weight matrix to remedy this. In terms of batch processing, the authors improve the use of ciphertext slots, which reduces memory use and computational complexity while improving inference efficiency. Another contribution is that the authors discovered that the homomorphic encryption scheme's parameter selection may somewhat modify the original picture, which can affect accuracy rates at the conclusion of the inference step. To address this, the authors propose a unique strategy in which matching noise is added during the training phase to make the model more resilient and adaptive to slightly "blurred" pictures, hence reducing the influence of encryption scheme noise during the inference stage.

A recent study of HE with applications in machine learning is Rovida [102]. The authors state that the goal of this study is to provide an efficient method capable of performing k-means training and classification on a set of encrypted data points using Euclidean distance based on the CKKS scheme [43]. A sequence of interactions between the server and the client take place during the training phase. In essence, the server may conduct a single loop and provide the result. This result is then processed by the client, who may request another iteration by communicating a new set of centroids. The client-performed data management step is faster than a traditional k-means execution. Each iteration, as will be shown later, needs a simple masking operation for each cluster, resulting in a complexity of $O(kn)$, where k represents the number of clusters and n represents the number of points (it is worth mentioning that the k operations may be parallelized). In contrast, converting an existing model into an encrypted model, as well as classifying an encrypted point, are easier operations that do not need several encounters.

A work that is applied in cybersecurity and employs ML and HE is Hu et al. [103]. This paper describes an intrusion detection approach that uses federated learning to exploit client-side security in Advanced Metering Infrastructure (AMI) networks, with the CKKS scheme used to protect model parameters. According to the authors, To prevent poisoning attacks in federated learning, the direction similarity between the model learned by the data processing center and the model trained by each client is originally determined. This similarity value is subsequently scaled to serve as the

aggregated model's adaptive weight. Following that, each client model update's size is normalized to match the size of the data processing center model update. Finally, the normalized updates and adaptive weights are weighted averaged to generate a global model update. The results of this study show that the suggested approach can successfully survive inference and poisoning attempts. The intrusion detection approach based on federated learning can maintain strong detection performance in the setting of the AMI network.

A work that uses PHE, namely Paillier cryptosystem is Fang and Qian [104]. The authors created a multi-party, privacy-protected machine learning system that combines homomorphic encryption with federated learning. During the training phase, this architecture maintains the security of the data and model. Furthermore, it protects private data during multi-party learning situations. The authors confirmed that the model trained using the suggested methodology outperforms models trained using standard methods. The accuracy deviation in the MNIST and metal fatigue datasets is $<1\%$, according to the experimental findings. The authors investigated the time overhead of homomorphic encryption, taking into account the effects of different key lengths and network architectures. The data show that when the key length or structural complexity rises, so does the time overhead. As a result, a careful balance between performance and security level is critical.

As we have seen, HE can be used in machine learning and artificial intelligence to enable secure data sharing and collaborative learning [105]. For instance, HE can be used to train machine learning models on encrypted medical data, such as electronic health records, while preserving patient privacy [106]. HE can also be used for secure multiparty computation (SMPC), which allows multiple parties to jointly compute on their data without revealing it, as in the case of federated learning [105]. HE can also be used in domains such as finance and nuclear safeguards, where privacy-preserving analytics are needed [106]. However, HE comes at the cost of efficiency and speed, and selecting the appropriate technique requires consideration of security requirements and potential limitations [106]. Specific examples of AI/ML areas in which HE can be applied are:

- Secure Machine Learning Prediction: Homomorphic encryption can be used to securely predict outcomes without revealing the data used to train the model. This can be applied in various domains, such as healthcare and finance, where privacy is crucial.
- Privacy-Preserving Data Sharing: Homomorphic encryption can be used to share data between organizations without revealing the actual data. This can be used in collaborative AI and machine learning projects.
- Encrypted Neural Networks: Homomorphic encryption can be used to train and execute neural networks on encrypted data. This can be used in applications where the data is highly sensitive, such as military and intelligence applications.

Figure 3.3 shows the main domains and industries related to machine learning and artificial intelligence where HE can be included.

However, the application of HE in ML and AI also presents challenges. The computational and storage overheads associated with HE can be substantial, impacting

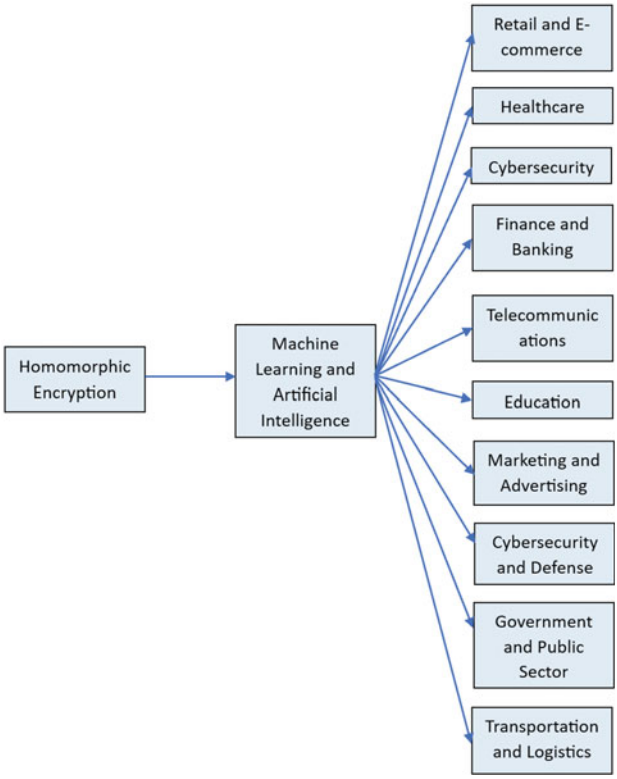

Fig. 3.3 Main domains and industries from machine learning and artificial intelligence where HE can be included

the efficiency of ML models. Furthermore, the choice of parameters for HE schemes can affect the accuracy of ML models, necessitating careful consideration and trade-offs between security, performance, and accuracy.

Despite these challenges, the potential of HE in ML and AI is immense. With ongoing research and development, it is anticipated that more efficient and practical HE schemes will be developed, further expanding the scope of privacy-preserving ML and AI applications. As the field continues to evolve, it will be crucial to develop standards and best practices for the use of HE in ML and AI, ensuring that these powerful technologies can be harnessed in a manner that respects and protects data privacy.

3.6.3 Blockchain

The property of performing computations on encrypted data is highly desirable in blockchain systems as it allows for secure and private transactions without revealing sensitive information to the network participants or the underlying blockchain infrastructure.

Homomorphic encryption is important in blockchain is its potential to enhance privacy and confidentiality. In traditional blockchain systems, all transaction data is typically transparent and accessible to all participants. While this transparency is important for the decentralized nature of blockchain, it can also be a challenge in scenarios where certain aspects of the transaction need to remain private, for example, in electronic vote systems, where the vote should be anonymous.

Additionally, homomorphic encryption can enhance the security of blockchain systems by mitigating the risk of data breaches. Since data is encrypted, even if an attacker gains unauthorized access to the blockchain or the underlying infrastructure, they would only be able to interact with encrypted data, which is computationally infeasible to decipher without the encryption key. This provides an extra layer of protection against data theft or unauthorized manipulation.

Also, homomorphic encryption enables secure and trustless computation on the blockchain. Smart contracts, which are self-executing contracts with predefined rules and conditions, are a fundamental aspect of blockchain technology. Homomorphic encryption allows for the execution of complex computations on encrypted data within smart contracts, ensuring that the privacy and confidentiality of the data are maintained throughout the computation process.

Only sharing the results of operational analysis of power data rather than the source data in the process of cross-agency flow of power data is an effective sharing method that can not only meet the needs of data value mining, but also prevent source data leakage and protect data rights and interests. Starting from this idea, the authors of Lin et al. [107] proposed a power data blockchain sharing scheme based on homomorphic encryption, which made data sharing rules and computing trustworthy with smart contracts, while also preventing data leaking with homomorphic encryption methods. Furthermore, a power data sharing platform was constructed and designed based on the sharing scheme, using Hyperledger Fabric[20] as the underlying blockchain architecture, so that the power data could be calculated realistically in the chain code Docker container environment. Finally, the authors emulated the sharing strategy using a two-node deployment on a single device. The response performance of the Hyperledger Caliper-tested sharing operation demonstrates the superiority of our architecture.

A large applicability of homomorphic encryption in the blockchain is e-volting. In the work [108], the authors have developed a secure online voting system that leverages the ElGamal Elliptic Curve Cryptosystem [4] for vote encryption. The additive homomorphic property of the Elliptic Curve Cryptosystem is employed to compute results without the need to decrypt each ballot individually. As an additional layer

[20] https://www.hyperledger.org/use/fabric.

of security, blockchain technology is used on the authority side to preserve ballots, with SHA-512 serving as the hashing mechanism. Another contribution related to e-voting is Salman et al. [109]. Here, the authors made three contributions to this study. To begin, they suggest a hybrid public key technique that encrypts voters' personal data using an elliptic curve while encrypting the vote and voter ID number with a homomorphic key. This enables the election administration to collect the encrypted ballots later. Second, they increase authenticity by including a QR code input method as well as a facial recognition stage. Finally, they provide a public key distribution and verification schema for validating the elliptic public key of the voting point and the homomorphic public key of the election authority. On the other hand, in Qu et al. [110] the authors propose a voting mechanism based on homomorphic encryption and blockchain technology to publish the voting process and replace the conventional trusted third party with a smart contract. It encrypts and signs the ballot using the homomorphic encryption and encryption algorithms, then uses their aggregation features to execute homomorphic tally on the encrypted votes. This method not only reduces the burden on voters, but it also improves voting efficiency. It also assures the security of computerized voting. Because of its low processing needs, it is more convenient and flexible for usage in large-scale voting situations.

Another study that involves HE in the blockchain is Zhou et al. [111]. Here, the authors present BeeKeeper, a unique blockchain-based threshold IoT service solution, in this study. Servers in the BeeKeeper system may process a user's data by conducting homomorphic calculations on it without learning anything about it. Furthermore, if both the node and the leader are willing, any node may become a leader's server. This enables BeeKeeper's performance to improve over time by attracting external computer resources. Furthermore, malicious nodes may be investigated. BeeKeeper is also fault-tolerant, as long as a certain number of its servers remain active and honest, a user's BeeKeeper protocol may run successfully. Finally, the authors implement BeeKeeper on the Ethereum network and offer performance results. In the testing, servers were able to respond in around 107 milliseconds. Furthermore, BeeKeeper's performance is mostly determined by the blockchain platform. For example, the response time is around 22.5 s owing to the Ethereum blockchain's block frequency of approximately 15 s. In reality, using a different blockchain with a shorter block period might drastically cut response time.

The use of homomorphic encryption in blockchain systems enhances privacy, security, and trust, making it a key component in ensuring the success and adoption of blockchain technology.

Figure 3.4 shows the main domains and industries related to blockchain where HE can be included.

Homomorphic encryption can enable more complex computations to be performed directly on the blockchain. This can open up new possibilities for decentralized applications (DApps) that require to process encrypted data.

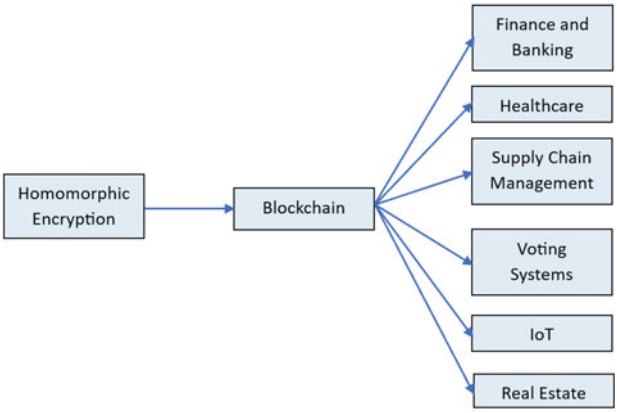

Fig. 3.4 Main domains and industries from blockchain where HE can be included

3.6.4 Internet of Things

Homomorphic encryption has the potential for enhancing security in the IoT, by ensuring that the data collected by the IoT devices remains encrypted while being processed, so privacy and confidentiality are preserved. However, the feasibility of implementing homomorphic encryption in IoT devices is a subject of ongoing research taking into consideration the context of IoT presented in Sect. 2.3.3. While IoT devices typically have limited computational power and storage, the current implementations of homomorphic encryption are computationally intensive, making it challenging to directly implement such encryption schemes on the devices themselves. A potential solution to this challenge is to integrate homomorphic encryption in the cloud-based components of IoT systems, such that IoT devices encrypt the data locally using less computationally intensive encryption schemes, then send it to the cloud. The cloud server can further perform the computationally intensive homomorphic operations, so that homomorphic encryption is implemented within the system without overburdening the IoT devices themselves. HE can be integrated with IoT systems (not necessarily implemented directly on the IoT devices) having the following advantages [112, 113]:

- Data Privacy: IoT devices often collect and transmit sensitive data. Homomorphic encryption allows for performing computations on encrypted data, ensuring that the data remains private and secure throughout the entire process. This is particularly important as IoT devices often operate in untrusted environments.
- Data Aggregation and Analytics: IoT devices generate vast amounts of data, and aggregating and analyzing this data is crucial for deriving meaningful insights. Homomorphic encryption enables secure and privacy-preserving data aggregation and analytics, allowing for valuable analysis while preserving the privacy of individual data points.

- Outsourcing Data Processing: Homomorphic encryption allows for secure outsourcing of data processing tasks to third-party service providers or cloud platforms, which means that IoT devices with limited computational resources to offload complex computations while maintaining data privacy.
- Trust and Security: IoT devices often interact with each other and with external systems. Homomorphic encryption helps establish trust and security by ensuring that sensitive data remains encrypted and protected during these interactions, reducing the risk of data breaches or unauthorized access.

In Trived et al. [114], the authors present a homomorphic cryptosystem-based secure data processing model (HC-SDPM) for safe data collecting and aggregation offloading on edge nodes for a suggested edge-assisted IoT healthcare system. To assure end-to-end secrecy and data integrity throughout transmission and aggregation, the authors use Paillier homomorphic encryption and certificateless linear homomorphic ID-based signatures. HC-SDPM also has a semi-trusted authority to allow resilience against key alterations. Under computational Diffie-Hellman assumptions, the security analysis of HC-SDPM provides signature unforgeability in the random oracle model. In terms of computational cost, communication overhead, and storage complexity, HC-SDPM outperforms similarly designed methods.

In work [115], the goal is to improve the security of information transfer, such as multimedia information generated by IoT devices. The Message Queue Telemetry Transport (MQTT) protocol is used over SSL/TLS in this scheme. Given the vulnerability of MQTT to eavesdropping, the authors propose the Elliptic curve-ElGamal encryption method, which offers a homomorphic factor and so mitigates man-in-the-middle assaults. The present study proposes dynamic key change and proportionate offloading of data to help save node energy by selectively sending data to the cloud and the fog depending on the data subject. The findings show that the system's security and longevity can be improved over current methods.

Figure 3.5 shows the main domains and industries related to IoT where HE can be included.

However, as we have seen in the previous sections, there are solutions based on blockchain and homomorphic encryption that addresses security issues in IoT systems [116–119].

3.7 Challenges and Research Directions

Fully Homomorphic Encryption has become an important paradigm in cryptography, which enables performing arbitrary computations on encrypted data without needing to decrypt it. Despite the significant theoretical and practical advances since its inception, several challenges are still opened. Below we list current challenges and research directions in FHE at the moment of writing this book.

1. Efficiency: One of the main challenges of FHE is the computational and storage efficiency. Compared to plain text operations, FHE schemes are very slow and

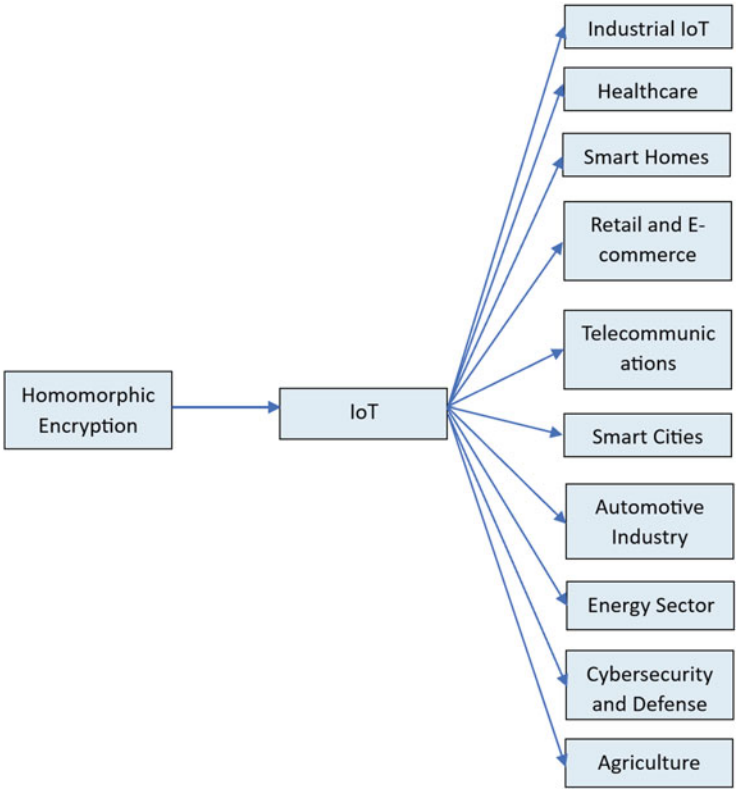

Fig. 3.5 Main domains and industries from IoT where HE can be included

require a significant amount of additional memory. This makes FHE impractical for most real-world applications at present. While there have been significant improvements in the efficiency of FHE schemes, there is still a long way to go before they become feasible for large-scale use. A research direction in this context would be to design more efficient FHE schemes that can process larger volumes of data more quickly and with less memory.

2. Bootstrapping: We have seen that bootstrapping is a process in FHE that allows for the reduction of noise in encrypted data, by "refreshing" the ciphertext. However, bootstrapping is a computationally intensive process that considerably slows down computations. Reducing the need for (or the cost of) bootstrapping represents an important research direction. Alternatives like leveled FHE, where the depth of circuits is fixed in advance and bootstrapping is not required, are being explored.
3. Practical Implementations: The practical implementation of FHE schemes generates many difficulties. It requires understanding of complex concepts and careful balancing of various parameters, such as the ciphertext noise and the security level. Moreover, it is challenging to ensure that the implementation is secure

against side-channel attacks, which are types of attacks that exploit information leaked during the computation process. Further research is needed to simplify the implementation process and enhance the security of implemented systems.
4. Standardization: Another challenge in FHE is standardization. Different FHE schemes have different properties, and it is challenging to compare them or to ensure their interoperability. The development of standards for FHE would facilitate its broader adoption and enable users to make informed decisions about which scheme to use. Standardization would also facilitate the development of best practices for implementation, enhancing the security and reliability of FHE systems.
5. Integration with ML and AI Applications: As ML and AI have continued to advance, the integration of FHE with these technologies has become a natural and significant research direction. This integration would allow for the training and evaluation of models on encrypted data, enhancing the privacy and security of machine learning systems. However, many machine learning algorithms are not easily compatible with FHE, and research is needed to develop techniques for adapting these algorithms for use with encrypted data.
6. Multiparty Computation: Multiparty computation (MPC) is a promising area where multiple parties can jointly compute a function over their inputs while keeping them private. The integration of FHE with MPC is another important research direction, allowing for more complex and flexible privacy-preserving computations. However, designing secure and efficient protocols for this integration presents a considerable challenge.
7. Post-quantum Security: Finally, as quantum computers advance, the security of current FHE schemes against quantum attacks is a matter of concern. Currently, lattice-based FHE schemes are considered to be resistant to quantum attacks. However, further research is needed to ensure the security of these schemes and to develop new FHE schemes that can resist against quantum attacks.

FHE remains a promising yet challenging field in cryptography. Despite its promising properties, the practical implementation of FHE is still faced with significant difficulties. The main challenges lie in the areas of computational efficiency, usability, and ensuring security in real-world applications. These obstacles prevent FHE from being widely adopted in real-world applications that require data privacy.

Research in FHE is focused on overcoming these challenges. Improving computational efficiency is mandatory, with efforts concentrated on optimization methods, reducing dimensionality, and simultaneous operations. In the realm of usability, developing user-friendly interfaces, abstracting complex operations, and ensuring compatibility with existing systems are research areas with the potential to bring FHE into the mainstream. Lastly, the need for rigorous security measures and standardization protocols for FHE are also important.

The future of FHE research is filled with exciting opportunities. If these challenges can be effectively addressed, FHE could revolutionize the way we handle data, providing robust data privacy without preventing computational processes. As we delve deeper into the era of big data, FHE could become an indispensable tool for cybersecurity, paving the way for a new age of secure data processing.

References

1. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. Found. Sec. Comput. **4**, 169–180 (1978)
2. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inform. Theory **22**, 644–654 (1976)
3. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**, 120–126 (1978)
4. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inform. Theory **31**, 469–472 (1985)
5. Gentry, C.: A Fully Homomorphic Encryption Scheme. Stanford University (2009)
6. Van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Advances in Cryptology—EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings, vol. 29, pp. 24–43. Springer (2010)
7. Armknecht, F., et al.: A guide to fully homomorphic encryption. Cryptology ePrint Archive (2015)
8. Halevi, S.: Homomorphic encryption. In: Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich, pp. 219–276. Springer (2017)
9. Goldwasser, S., Micali, S.: Probabilistic encryption & how to play mental poker keeping secret all partial information. In: Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali, pp. 173–201 (2019)
10. Rothblum, R.: Homomorphic encryption: from private-key to public-key. In: Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28–30, 2011. Proceedings, vol. 8, pp. 219–234. Springer (2011)
11. Lindell, Y.: How to simulate it—a tutorial on the simulation proof technique. In: Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich, pp. 277–346 (2017)
12. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999, Proceedings, vol. 18, pp. 223–238. Springer (1999)
13. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. SIAM J. Comput. **43**, 831–871 (2014)
14. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. ACM Trans. Comput. Theory (TOCT) **6**, 1–36 (2014)
15. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Advances in Cryptology—CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2012. Proceedings, pp. 868–886. Springer (2012)
16. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive (2012)
17. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Advances in Cryptology—ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3–7, 2017, Proceedings, Part I, vol. 23, pp. 409–437. Springer (2017)

18. Marcolla, C., et al.: Survey on fully homomorphic encryption, theory, and applications. Proc. IEEE **110**, 1572–1609 (2022)
19. Silverberg, A.: Fully homomorphic encryption for mathematicians. Women in Numbers 2: Research Directions in Number Theory, vol. 606, p. 111 (2013)
20. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Cryptogr. **71**, 57–81 (2014)
21. Gentry, C., Halevi, S.: Implementing Gentry's fully-homomorphic encryption scheme. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 129–148. Springer (2011)
22. Scholl, P., Smart, N.P.: Improved key generation for Gentry's fully homomorphic encryption scheme. In: IMA International Conference on Cryptography and Coding, pp. 10–22. Springer (2011)
23. Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Advances in Cryptology—EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15–19, 2011. Proceedings, vol. 30, pp. 27–47. Springer (2011)
24. Coron, J.-S., Mandal, A., Naccache, D., Tibouchi, M.: Fully homomorphic encryption over the integers with shorter public keys. In: Annual Cryptology Conference, pp. 487–504. Springer (2011)
25. Cheon, J.H., et al.: Batch fully homomorphic encryption over the integers. In: Advances in Cryptology—EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26–30, 2013. Proceedings, vol. 32, pp. 315–335. Springer (2013)
26. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Math. Cryptol. **9**, 169–203 (2015)
27. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology—CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2013. Proceedings, Part I, pp. 75–92. Springer (2013)
28. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Annual Cryptology Conference, pp. 505–524. Springer (2011)
29. Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 617–635. Springer (2009)
30. Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In: Cryptography and Coding: 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17–19, 2013. Proceedings, vol. 14, pp. 45–64. Springer (2013)
31. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Annual Cryptology Conference, pp. 850–867. Springer (2012)
32. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In: Advances in Cryptology—CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17–21, 2014. Proceedings, Part I, vol. 34, pp. 297–314. Springer (2014)
33. Bajard, J.-C., Eynard, J., Hasan, M.A., Zucca, V.: A full RNS variant of FV like somewhat homomorphic encryption schemes. In: International Conference on Selected Areas in Cryptography, pp. 423–442. Springer (2016)
34. Halevi, S., Polyakov, Y., Shoup, V.: An improved RNS variant of the BFV homomorphic encryption scheme. In: Topics in Cryptology-CT-RSA 2019: The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4–8, 2019. Proceedings, pp. 83–105. Springer (2019)
35. Bajard, J.C., Eynard, J., Martins, P., Sousa, L., Zucca, V.: Note on the noise growth of the RNS variants of the BFV scheme. Cryptology ePrint Archive (2019)

36. Chen, H., Han, K.: Homomorphic lower digits removal and improved FHE bootstrapping. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 315–337. Springer (2018)
37. Brakerski, Z.: Fundamentals of fully homomorphic encryption–a survey. Electron. Colloquium Comput. Complex. **25**, 125 (2018)
38. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 617–640. Springer (2015)
39. Chillotti, I., Gama, N., Georgieva, M., Izabachene, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. in Advances in Cryptology—ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4–8, 2016. Proceedings, Part I, vol. 22, pp. 3–33. Springer (2016)
40. Chen, H., Chillotti, I., Song, Y.: Improved bootstrapping for approximate homomorphic encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 34–54. Springer (2019)
41. Kim, A., et al.: General bootstrapping approach for RLWE-based homomorphic encryption. Cryptology ePrint Archive (2021)
42. Lee, Y., et al.: Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 227–256. Springer (2023)
43. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: Advances in Cryptology—EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29–May 3, 2018. Proceedings, Part I, vol. 37, pp. 360–384. Springer (2018)
44. Kim, D., Song, Y.: Approximate Homomorphic Encryption over the Conjugate-Invariant Ring. In: Lee, K. (eds.) Information Security and Cryptology-ICISC 2018, pp. 85–102. Springer International Publishing (2019)
45. Kim, A., Papadimitriou, A., Polyakov, Y.: Approximate homomorphic encryption with reduced approximation error. In: Galbraith, S.D. (ed.) Topics in Cryptology—CT-RSA 2022, pp. 120–144. Springer International Publishing (2022)
46. Boemer, F., Costache, A., Cammarota, R., Wierzynski, C.: nGraph-HE2: a high-throughput framework for neural network inference on encrypted data. In: Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pp. 45–56. Association for Computing Machinery (2019)
47. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: a ring-based public key cryptosystem. In: Buhler, J.P. (ed.) Algorithmic Number Theory, pp. 267–288. Springer (1998)
48. Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Paterson, K.G. (ed.) Advances in Cryptology—EUROCRYPT 2011, pp. 27–47. Springer (2011)
49. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing, pp. 1219–1234. Association for Computing Machinery (2012)
50. Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In: Stam, M. (ed) Cryptography and Coding, pp. 45–64. Springer (2013)
51. Doróz, Y., Sunar, B.: Flattening NTRU for evaluation key free homomorphic encryption. J. Math. Cryptol. **14**, 66–83 (2020)
52. Costache, A., Laine, K., Player, R.: Evaluating the effectiveness of heuristic worst-case noise analysis in FHE. In: Chen, L., Li, N., Liang, K., Schneider, S. (eds.) Computer Security-ESORICS 2020, pp. 546–565. Springer International Publishing (2020)
53. Cheon, J.H., Jeong, J., Lee, C.: An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without a low-level encoding of zero. LMS J. Comput. Math. **19**, 255–266 (2016)

54. Albrecht, M., et al.: Homomorphic encryption standard. In: Protecting Privacy Through Homomorphic Encryption, pp. 31–62 (2021)
55. Computer Security Division, I. T. L. Post-quantum Cryptography|CSRC|CSRC. CSRC|NIST (2017). https://csrc.nist.gov/projects/post-quantum-cryptography
56. ISO/IEC WD 18033-8. ISO. https://www.iso.org/standard/83139.html
57. Chillotti, I., Gama, N., Georgieva, M., Izabachéne, M.: TFHE: fast fully homomorphic encryption over the torus. J. Cryptol. **33**, 34–91 (2020)
58. Homomorphic Encryption Standardization—An Open Industry/Government/Academic Consortium to Advance Secure Computation. https://homomorphicencryption.org/
59. Gouert, C., Mouris, D., Tsoutsos, N.G.: New insights into fully homomorphic encryption libraries via standardized benchmarks. Cryptology ePrint Archive (2022)
60. Doan, T. V. T., Messai, M.-L., Gavin, G., Darmont, J.: A survey on implementations of homomorphic encryption schemes. J. Supercomput. 1–42 (2023)
61. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Advances in Cryptology—CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2012. Proceedings, pp. 850–867. Springer (2012)
62. Al Badawi, A., et al.: Openfhe: open-source fully homomorphic encryption library. In: Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pp. 53–63 (2022)
63. Doróz, Y., Öztürk, E., Sunar, B.: Accelerating fully homomorphic encryption in hardware. IEEE Trans. Comput. **64**, 1509–1521 (2014)
64. Yang, Z., Hu, S., Chen, K.: FPGA-based hardware accelerator of homomorphic encryption for efficient federated learning (2020). arXiv preprint arXiv:2007.10560
65. Su, Y., Yang, B., Yang, C., Tian, L.: FPGA-based hardware accelerator for leveled ring-LWE fully homomorphic encryption. IEEE Access **8**, 168008–168025 (2020)
66. Geelen, R., et al.: BASALISC: Flexible asynchronous hardware accelerator for fully homomorphic encryption (2022). arXiv preprint arXiv:2205.14017
67. Mert, A. C., Öztürk, E., Savaş, E.: Design and implementation of encryption/decryption architectures for BFV homomorphic encryption scheme. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **28**, 353–362 (2019)
68. Agrawal, R., Bu, L., Ehret, A., Kinsy, M.A.: Fast arithmetic hardware library for RLWE-based homomorphic encryption (2020). arXiv preprint arXiv:2007.01648
69. Paul, B., Yadav, T.K., Singh, B., Krishnaswamy, S., Trivedi, G.: A resource efficient software-hardware co-design of lattice-based homomorphic encryption scheme on the FPGA. IEEE Trans. Comput. **72**, 1247–1260 (2022)
70. Cousins, D.B., et al.: TREBUCHET: fully homomorphic encryption accelerator for deep computation (2023). https://eprint.iacr.org/2023/521
71. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 124–134 (1994)
72. Wolf, C., Preneel, B.: Asymmetric cryptography: hidden field equations (2004). https://eprint.iacr.org/2004/072
73. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: an efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) Advances in Cryptology—ASIACRYPT 2018, pp. 395–427. Springer International Publishing (2018)
74. Shahid, F., Khan, A., Malik, S.U.R., Choo, K.-K.R.: WOTS-S: a quantum secure compact signature scheme for distributed ledger. Inform. Sci. **539**, 229–249 (2020)
75. Kaye, P., Laflamme, R., Mosca, M.: An Introduction to Quantum Computing. Oxford University Press Inc. (2007)
76. LaPierre, R.: Introduction to Quantum Computing. Springer International Publishing (2021)
77. Broadbent, A., Jeffery, S.: Quantum homomorphic encryption for circuits of low T-gate complexity. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology—CRYPTO 2015, pp. 609–629. Springer (2015)
78. Zhang, J.-W., Xu, G., Chen, X.-B., Chang, Y., Dong, Z.-C.: Improved multiparty quantum private comparison based on quantum homomorphic encryption. Physica A: Stat. Mech. Appl. **610**, 128397 (2023)

79. Tan, S.-H., Kettlewell, J.A., Ouyang, Y., Chen, L., Fitzsimons, J.F.: A quantum approach to homomorphic encryption. Sci. Rep. **6**, 33467 (2016)
80. Broadbent, A., Jeffery, S.: Quantum homomorphic encryption for circuits of low T-gate complexity. In: Annual Cryptology Conference, pp. 609–629. Springer (2015)
81. Ambainis, A., Mosca, M., Tapp, A., De Wolf, R.: Private quantum channels. In: Proceedings 41st Annual Symposium on Foundations of Computer Science, pp. 547–553. IEEE (2000)
82. Yu, L., Pérez-Delgado, C.A., Fitzsimons, J.F.: Limitations on information-theoretically-secure quantum homomorphic encryption. Phys. Rev. A **90**, 050303 (2014)
83. Liang, M.: Quantum fully homomorphic encryption scheme based on universal quantum circuit. Quant. Inform. Process. **14**, 2749–2759 (2015)
84. Dulek, Y., Schaffner, C., Speelman, F.: Quantum homomorphic encryption for polynomial-sized circuits. In: Advances in Cryptology—CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14–18, 2016. Proceedings, Part III, vol. 36, pp. 3–32. Springer (2016)
85. Ouyang, Y., Tan, S.-H., Fitzsimons, J.F.: Quantum homomorphic encryption from quantum codes. Phys. Rev. A **98**, 042334 (2018)
86. Cramer, R., Damgård, I.B.: Secure multiparty computation. Cambridge University Press (2015)
87. Franklin, M., Haber, S.: Joint encryption and message-efficient secure computation. J. Cryptol. **9**, 217–232 (1996)
88. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. In: Pfitzmann, B. (ed.) Advances in Cryptology—EUROCRYPT 2001, pp. 280–300. Springer (2001)
89. Zheng, W., Popa, R.A., Gonzalez, J.E., Stoica, I.: Helen: maliciously secure coopetitive learning for linear models. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 724–738 (2019)
90. Froelicher, D., Troncoso-Pastoriza, J.R., Sousa, J.S., Hubaux, J.-P.: Drynx: decentralized, secure, verifiable system for statistical queries and machine learning on distributed datasets. IEEE Trans. Inform. Forensics Secru. **15**, 3035–3050 (2020)
91. Asharov, G., et al.: Multiparty computation with low communication, computation and interaction via threshold FHE. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology-EUROCRYPT 2012, pp. 483–501. Springer (2012)
92. Lopez-Alt, A., Tromer, E., Vaikuntanathan, V.: Cloud-assisted multiparty computation from fully homomorphic encryption (2011). https://eprint.iacr.org/2011/663
93. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**, 34:1–34:40 (2009)
94. Maurer, U.: Unifying zero-knowledge proofs of knowledge. In: International Conference on Cryptology in Africa, pp. 272–286. Springer (2009)
95. Gentry, C., et al.: Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. J. Cryptol. **28**, 820–843 (2015)
96. Cramer, R., Damgård, I.: Zero-knowledge proofs for finite field arithmetic, or: can zero-knowledge be for free? In: Advances in Cryptology—CRYPTO'98: 18th Annual International Cryptology Conference Santa Barbara, California, USA, August 23–27, 1998. Proceedings, vol. 18, pp. 424–441. Springer (1998)
97. Steffen, S., Bichsel, B., Baumgartner, R., Vechev, M.: Zeestar: private smart contracts by homomorphic encryption and zero-knowledge proofs. In: 2022 IEEE Symposium on Security and Privacy (SP), pp. 179–197. IEEE (2022)
98. Kiesel, R., et al.: Potential of homomorphic encryption for cloud computing use cases in manufacturing. J. Cybersecur. Privacy **3**, 44–60 (2023)
99. Munjal, K., Bhatia, R.: A systematic review of homomorphic encryption and its contributions in healthcare industry. Complex Intell, Syst (2022)
100. Ibtihal, M., Driss, E.O., Hassan, N.: Homomorphic encryption as a service for outsourced images in mobile cloud computing environment. In: Cryptography: breakthroughs in Research and Practice, pp. 316–330. IGI Global (2020)

101. Song, C., Huang, R.: Secure convolution neural network inference based on homomorphic encryption. Appl. Sci. **13** (2023)
102. Rovida, L.: Fast but approximate homomorphic k-means based on masking technique. Int. J. Inform, Secur (2023)
103. Hu, C., Yu, F., Wang, J., Chen, Y., Xia, Z.: Intrusion detection framework based on homomorphic encryption in AMI network. Front. Phys. **10** (2022)
104. Fang, H., Qian, Q.: Privacy preserving machine learning with homomorphic encryption and federated learning. Fut. Internet **13**, 94 (2021)
105. Lauter, K.: Private AI: machine learning on encrypted data. In: Chacón Rebollo, T., Donat, R., Higueras, I. (eds.) Recent Advances in Industrial and Applied Mathematics, pp. 97–113. Springer International Publishing, (2022)
106. Arnold, D., Saniie, J., Heifetz, A.: Homomorphic encryption for machine learning and artificial intelligence applications (2022). https://www.osti.gov/biblio/1886256
107. Lin, Y., et al.: Power data blockchain sharing scheme based on homomorphic encryption. In: 2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), vol. 5, pp. 625–629 (2022)
108. Chandra Priya, J., Sathia Bhama, P. R. K., Swarnalaxmi, S., Aisathul Safa, A., Elakkiya, I.: Blockchain centered homomorphic encryption: a secure solution for e-balloting. In: Pandian, A.P., Senjyu, T., Islam, S.M.S., Wang, H. (eds.) Proceeding of the International Conference on Computer Networks, Big Data and IoT (ICCBI—2018), pp. 811–819. Springer International Publishing (2020)
109. Salman, S.A.B., Al-Janabi, S., Sagheer, A.M.: Valid blockchain-based e-voting using elliptic curve and homomorphic encryption. Int. J. Interac. Mob. Technol. **16**(20) (2022)
110. Qu, W., Wu, L., Wang, W., Liu, Z., Wang, H.: A electronic voting protocol based on blockchain and homomorphic signcryption. Concurr. Comput.: Pract. Exp. **34**, e5817 (2022)
111. Zhou, L., Wang, L., Sun, Y., Lv, P.: BeeKeeper: a blockchain-based iot system with secure storage and homomorphic computation. IEEE Access **6**, 43472–43488 (2018)
112. Shrestha, R., Kim, S.: Integration of IoT with blockchain and homomorphic encryption: challenging issues and opportunities. Adv. Comput. **115**, 293–331 (2019)
113. Dorević, G., Marković, M., Vuletić, P.: Evaulation of homomorphic encryption implementation on Iot device. JITA—J. Inform. Technol. Appl. (Banja Luka)—APEIRON **23** (2022)
114. Trivedi, H.S., Patel, S.J.: Homomorphic cryptosystem-based secure data processing model for edge-assisted IoT healthcare systems. Internet Things **22**, 100693 (2023)
115. Gupta, S., et al.: Energy-efficient dynamic homomorphic security scheme for fog computing in IoT networks. J. Inform. Secur. Appl. **58**, 102768 (2021)
116. Yi, H., et al.: Energy trading IoT system based on blockchain. Swarm Evol. Comput. **64**, 100891 (2021)
117. Xu, G., Zhang, J., Wang, L.: An edge computing data privacy-preserving scheme based on blockchain and homomorphic encryption. In: 2022 International Conference on Blockchain Technology and Information Security (ICBCTIS), pp. 156–159 (2022)
118. She, W., et al.: Homomorphic consortium blockchain for smart home system sensitive data privacy preserving. IEEE Access **7**, 62058–62070 (2019)
119. Loukil, F., Ghedira-Guegan, C., Boukadi, K., Benharkat, A.-N.: Privacy-preserving IoT data aggregation based on blockchain and homomorphic encryption. Sensors **21**, 2452 (2021)

Chapter 4
Searchable Encryption

4.1 History

Searchable encryption (SE) is a prominent field within contemporary cryptography. It allows users to conduct specific searches over encrypted data, offering several benefits. Firstly, it enables data to be stored in its encrypted form on third-party or cloud servers, enhancing data security. Secondly, the use of search indexes simplifies the search process, as the encrypted data can remain on the server, eliminating the need to download it for searching. Lastly, it ensures that users submitting the search query cannot access the raw, unencrypted data during the search operation, further safeguarding data privacy.

The first mention of searchable encryption dates back to the early 2000s. The concept was first introduced in [1]. The authors proposed a symmetric searchable encryption (SSE) scheme that allowed keyword search over encrypted data, marking a significant milestone in the field.

The following years saw a flurry of research activity, with the introduction of public-key searchable encryption (PEKS) in [2]. This work expanded the possibilities of searchable encryption, allowing anyone with a public key to encrypt data, but only those with the corresponding private key could search it.

The field continued to evolve with the introduction of Order-Preserving Symmetric Encryption (OPSE) in [3]. This technique allowed for the comparison of encrypted data, opening up new possibilities for database queries on encrypted data.

In 2010, in [4] was proposed a new model for SSE that supported dynamic operations on the encrypted data. This work laid the foundation for many of the dynamic SSE schemes we see today.

Despite these advancements, challenges remain. The balance between search efficiency and security is a constant theme in the literature. Works, such as [5] (encryption scheme called *Sophos*), have proposed new constructions that aim to balance these competing demands.

S. L. Nita and M. I. Mihailescu, *Advances to Homomorphic and Searchable Encryption*,
https://doi.org/10.1007/978-3-031-43214-9_4

In the last years, the research literature has introduced a combination between searchable encryption and homomorphic encryption.

4.2 Components

This section is dedicated to discussing the key entities involved in searchable encryption schemes and the architectural design that supports these systems. The primary entities typically include the data owner, who encrypts and uploads data to the server; the server, which stores the encrypted data and performs search operations; and the data user, who submits encrypted search queries and retrieves relevant data.

The architecture of searchable encryption schemes, on the other hand, is generally composed of two main modules: the setup and the retrieval. The setup involves processes such as key generation, data encryption, and index creation, while the retrieval encompasses search query submission, search operation execution, and data decryption.

4.2.1 Entities

In searchable encryption schemes, there are several well-established entities that interact with the algorithms. Following this interaction, the system should ensure secure and efficient search operations over encrypted data. Further, we will overview these entities, describing their roles, responsibilities, and the interactions among them within a typical SE scheme. Below, we describe the parties that are working with data in the searchable encryption schemes based on works [6, 7]:

- *The data owner* (DO) is an entity that possesses a collection of documents organized as the set $D = \{D_1, \ldots, D_n\}$, where n represents the total number of documents. Each document D_i (with i ranging from 1 to n) is associated with a set of keywords $K_i = \{k_i^{i_1} \ldots k_i^{i_j}\}$, where j denotes the number of keywords defining that particular document. Additionally, the data owner is responsible for generating encryption keys to secure both the set of documents and the associated sets of keywords. These data and keywords are then encrypted by the data owner. Subsequently, the data owner sends the encrypted data to a designated server for storage. Furthermore, the data owner establishes various rights and access policies for different types of users.
- *The data user* (DU) is an entity that can submit search queries to the server and receive the search results. When searching among the encrypted documents, the data owner generates a *trapdoor* value based on a specific keyword and submits it to the server. The trapdoor encodes the data user's search criteria. The search results are then received by the data user as either a set of documents or a set of

distinct identifiers correlated with the documents. It is important to note that the data owner can also submit search queries and act as a data user simultaneously [8–11].
- *The (Cloud) Server* (CS) is the entity responsible for storing data in its encrypted state, receiving trapdoors, and conducting searches within the encrypted documents. It is commonly configured in an *honest-but-curious* model.
- *Trusted Authority* (TA) is an often present entity that generates the keys instead of the DO and applies security policies if included. In early versions of SE, this entity was not included, but nowadays it is widely used.
- *The Trusted computing base (Gateway)* is an entity that can be considered as part of the previous entities, although it is sometimes treated separately. The trusted computing base is responsible for pre-processing the data, which can be done either on the user's device or on a trusted server. One example of pre-processing is the removal of stop keywords from the search query, as mentioned in the work [12].

The searchable encryption schemes can be constructed from Symmetric Cryptography as well as Public-Key Cryptography, as we will see below. An SE scheme consists of several probabilistic polynomial-time algorithms such as: `KeyGeneration`, `Encryption`, `BuildIndex`, `Trapdoor`, `Search`, `Decryption`. Each entity described here triggers specific algorithms. Figure 4.1 shows a general overview of the entities involved in a SE system. The red arrows show actions triggered by the trusted authority, the blue arrows show properties or actions related to the data owner, the green arrows show actions related to data users, and, lastly, the black arrows show actions made by the server.

4.2.2 Architecture

An important factor to consider from the beginning is the selection of the algorithms from a searchable encryption scheme for implementation within a cloud computing environment, taking into account the physical architecture of the system. It's worth noting that some searchable encryption algorithms are executed on a single machine, a practice that may not be ideal from a security standpoint. By establishing a robust cloud architecture where multiple machines can be adequately secured, it becomes feasible to assign a dedicated machine (such as a server) to each searchable encryption algorithm.

The structure of searchable encryption schemes comprises two essential modules: *the setup* and *the retrieval*.

The setup module

The setup module contains procedures such as pre-processing of documents, key generation, encryption, and storing encrypted documents on cloud servers. In the

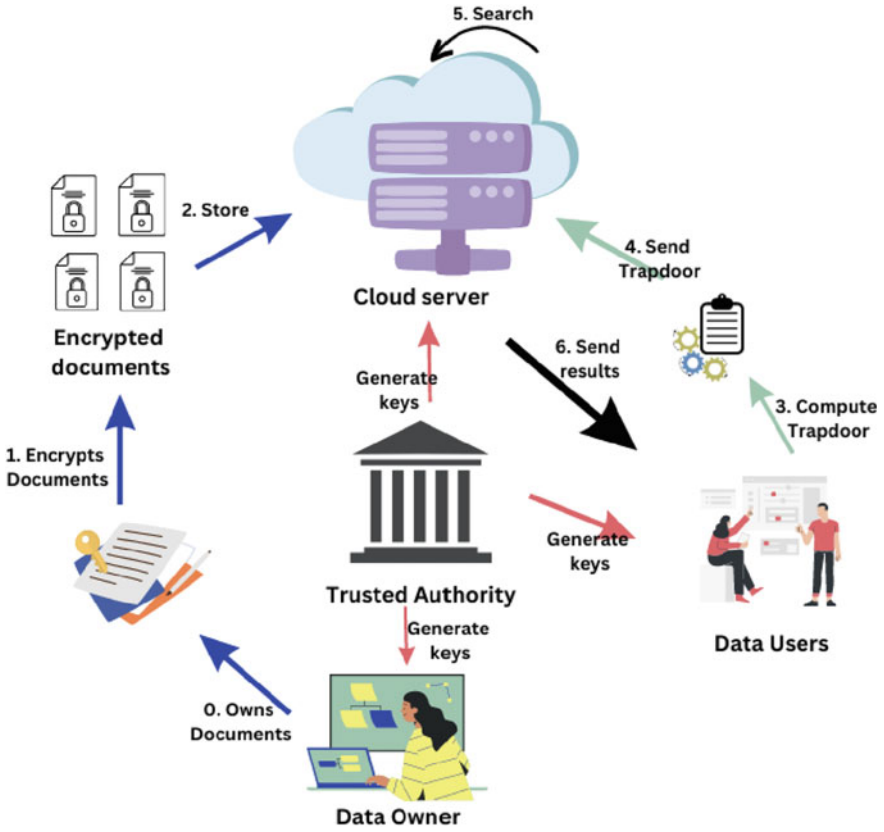

Fig. 4.1 Entities of a general SE

works of [13, 14], the authors suggest pre-processing operations that extract information describing the documents in relation to the type of search. The subsequent algorithm, key generation, is designed to generate the encryption system's keys: a secret key used for both encryption and decryption in a symmetric scheme, and a pair of secret (private) and public keys in a public scheme, where the public key is used for encryption and the private key for decryption. The setup module may also include optional algorithms, such as the generation of an index structure for documents [12–14], or a metadata structure [15], or a keywords structure [13, 16]. Subsequently, the data is encrypted (both the documents and the generated structure), with the documents being encrypted into a searchable encrypted document [1]. The final procedure involves the data owner sending the encrypted data to the cloud servers and assigning access rights to different types of users.

The retrieval module

The retrieval module includes procedures such as search query submission, search operation, and document retrieval based on the search rule, matching the trapdoor submitted by the data user. To search for documents containing specific keywords, the data user selects a set of desired keywords $W = w_1, \ldots, w_k$ and generates a trapdoor value T_W based on these. The server uses the trapdoor value to perform matching tests with the index structure. An optional procedure in this module is search query pre-processing [8, 17]. Upon receiving the encrypted documents from the server, the data user decrypts them using the private key.

Types of FE schemes

Following the two main branches of cryptography, namely Symmetric Cryptography and Asymmetric Cryptography (or Public-Key Cryptography), searchable encryption schemes also fall into one of these categories. This classification is not merely a theoretical construct but has significant practical implications, particularly in terms of security, efficiency, and the types of applications they are best suited for. Therefore, from this perspective, there are two categories of searchable encryption schemes: Symmetric Searchable Encryption (SSE) and Public-Key Searchable Encryption (PKSE). SSE and PKSE represent two distinct approaches to the challenge of performing search operations on encrypted data, each with its advantages and disadvantages.

SSE is known for its efficiency and simplicity, making it an ideal choice for scenarios where speed and resource usage are critical. However, it operates on symmetric key principles, meaning the same key is used for both encryption and decryption. This can limit its utility in scenarios where data needs to be shared securely among multiple parties.

On the other hand, PKSE operates on public key principles, providing a higher level of security and making it possible to securely share data among multiple parties, where each party has a pair of keys—a public key for encryption and a private key for decryption. This allows for more complex and secure data sharing scenarios, but at the cost of increased computational complexity and resource usage.

Further, we will present each of these categories.

4.2.2.1 Symmetric Searchable Encryption

The following algorithms are included in symmetric searchable encryption schemes [7]:

- `KeyGeneration`$(\lambda) \rightarrow key_{secret}$: This algorithm takes the security parameter λ as input and outputs the secret key key_{secret}. This algorithm is executed by the data owner.

- $\texttt{Encryption}(key_{secret}, D) \rightarrow C$: This algorithm takes the private key key_{secret} and the document set D as input, and outputs the corresponding set of encrypted documents C. This algorithm is executed by the data owner.
- $\texttt{BuildIndex}(key_{secret}, D) \rightarrow I_D$: This algorithm takes the secret key key_{secret} and the document set D as input, and outputs the index structure I_D. This algorithm is executed by the data owner.
- $\texttt{Trapdoor}(key_{secret}, w) \rightarrow T_w$: This algorithm takes the secret key key_{secret} and a specific keyword w as input, and outputs the trapdoor value T_w. This algorithm is executed by the data user.
- $\texttt{Search}(I_D, T_w) \rightarrow C_w$: This algorithm takes the index structure I_D and the trapdoor T_w as input, and outputs the encrypted documents $C_w = \{C_w^1 \dots C_w^m\}$ that match with the trapdoor value. This algorithm is executed by the server.
- $\texttt{Decryption}(key_{secret}, C_w) \rightarrow D_w$: This algorithm takes the set of encrypted documents C_w sent by the server and the secret key key_{secret} as input, and outputs the set of decrypted documents D_w. This algorithm is executed by the data user.

Figure 4.2 shows the sequence diagram for the SSE schemes, highlighting the algorithms included in the setup module, and retrieval module, respectively. Note that, the algorithms of the setup module are marked with blue, while the algorithms from the retrieval module are marked with green.

A special type of SSE is Dynamic Symmetric Searchable Encryption (DSE or DSSE). We will talk in detail about it in Sect. 4.5.2.

4.2.2.2 Public-Key Searchable Encryption

The first scheme within this category is the *Public Key Encryption with Keyword Search* (PEKS) [2] scheme, which contains four algorithms, detailed as follows:

- $\texttt{KeyGeneration}(\lambda) \rightarrow (key_{secret}, key_{public})$: This algorithm takes the security parameter λ as input and outputs the pair of keys key_{secret} and key_{public}.
- $\texttt{PEKS}(key_{public}, kw) \rightarrow c$: This encryption algorithm takes the public key key_{public} and the keyword kw as input, and generates c, the encrypted form of kw.
- $\texttt{Trapdoor}(key_{secret}, w) \rightarrow T_w$: This algorithm takes the secret key key_{secret} and the query keyword w as input, and outputs the trapdoor value T_w.
- $\texttt{Test}(key_{public}, c', T_w) \rightarrow 0, 1$: This algorithm takes the public key key_{public}, the encrypted value c' of a keyword w', and the trapdoor value T_w as input. It outputs 1 (i.e., `true`) if $w = w'$ and 0 (i.e., `false`) otherwise.

Definition 4.1 The **consistency of the PEKS** scheme requires the fulfillment of the following rule: for any keyword kw and for any pair $(key_{secret}, key_{public}) \leftarrow \texttt{KeyGeneration}(\lambda)$, and for any trapdoor value $T_{kw} \leftarrow \texttt{Trapdoor}(key_{secret}, kw)$, the below condition is satisfied:

$$\texttt{Test}(key_{public}, \texttt{PEKS}(key_{public}, kw), T_{kw}) = 1 \tag{4.1}$$

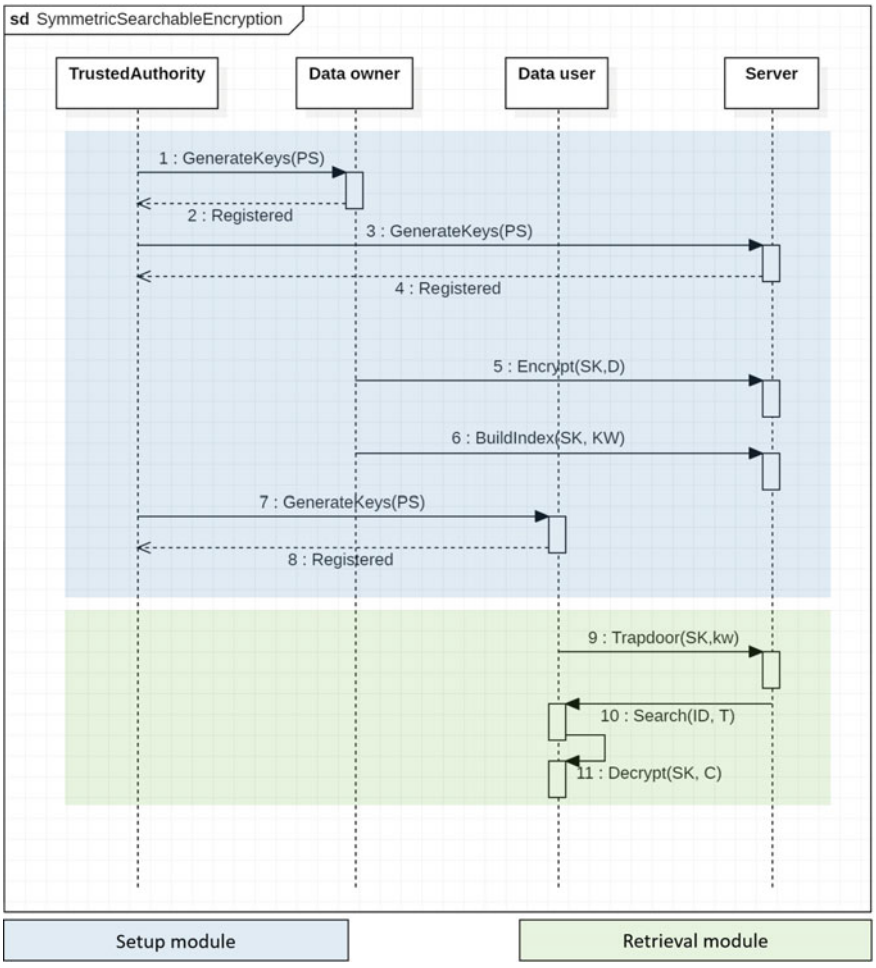

Fig. 4.2 Sequence diagram for SSE

Another significant contribution to public key searchable encryption is presented in [18].

Definition 4.1 implies that the `Test` algorithm must identify any valid trapdoor value.

Following the introduction of the first PEKS scheme [2], numerous variations have been proposed in the literature. Further, we present a form of the PEKS scheme that contains the same algorithms as the symmetric schemes, albeit with some modifications to the parameters, as detailed in [7]:

- `KeyGeneration`$(\lambda) \rightarrow (key_{secret}, key_{public})$: The output of the key generation algorithm differs, resulting in a pair of keys—the public and the secret key.

- $\texttt{Encryption}(key_{public}, D) \rightarrow C$: The input of the encryption algorithm has changed, with the public key now being used for encryption.
- $\texttt{BuildIndex}(key_{public}, D_i, w_{D_i}) \rightarrow I_{D_i}$: The input of the build index algorithm has changed. The public key is now used to construct the index structure for the set of documents.
- $\texttt{Trapdoor}(key_{secret}, w) \rightarrow T_w$: The input and output of the trapdoor algorithm remain consistent with those of symmetric searchable encryption schemes.
- $\texttt{Search}(key_{public}, I_D, T_w) \rightarrow C_w$: The search algorithm includes an additional parameter, specifically the public key.
- $\texttt{Decryption}(key_{secret}, C_w) \rightarrow D_w$: The decryption algorithm maintains the same input and output as in the symmetric searchable encryption schemes.

Given these modifications, the definition of consistency is changed as follows:

Definition 4.2 For any keyword kw, there exists an encrypted keyword C_{kw} such that for $I_D \leftarrow \texttt{BuildIndex}$, for any pair $(key_{secret}, key_{public}) \leftarrow \texttt{KeyGeneration}(\lambda)$, and for any trapdoor value $T_{kw} \leftarrow \texttt{Trapdoor}(key_{secret}, kw)$, the following condition is satisfied:

$$\texttt{Search}((key_{public}, I_D, kw), T_{kw}) = C_{kw} \tag{4.2}$$

Please note that we may occasionally refer to the consistency property as the **correctness** of the search/test algorithm.

4.3 Security

4.3.1 Security Requirements

During the search process or from the encrypted data, the server should not learn any information about the plain data. To clarify this, the authors of [1] outlined the following rules that searchable encryption schemes should adhere to:

- *Controlled searching*: Search queries should not be submitted by unauthorized users. In [19, 20], the authors proposed searchable encryption schemes in which queries can only be submitted by a user if they possess the appropriate key to generate trapdoor values. At the same time, on the server-side, the data must be encrypted and operations must be performed over encrypted data.
- *Encrypted queries*: Search queries should be encrypted before being submitted to the server. This ensures that the query results will not reveal any information to an untrusted server. The authors of [19] proposed a scheme in which they used secure indexes for the search query.
- *Query isolation*: The search process should not reveal any information to the server. However, the server should be able to send the results of the search query to the data user.

Furthermore, searchable encryption schemes need to ensure security for the following patterns:

- *Search pattern*: This refers to any information that can be learned from the fact that two distinct results are derived from the same keyword. Several initial works [1, 11, 19] prioritized efficiency over the search pattern.
- *Access pattern*: This refers to any information that can be learned from the fact that a set of documents $D^w = \{D_1^w, \ldots, D_k^w\}$ are characterized by the same keyword w. To hide the access pattern, the authors of [21] employ Oblivious RAM (ORAM), which repeatedly shuffles the data blocks. With this technique, the same search query result is accessed through different memory blocks.

4.3.2 Security Models

4.3.2.1 Security Models for SSE

Initial security models for symmetric searchable encryption schemes are introduced in [19]. In this work, the authors introduce IND-CKA1 (Chosen Keyword Attack for indexes) and the stronger model IND-CKA2, both of which are focused on the security of the index structure. These security models ensure that an unauthorized user who analyzes the secure indexes gains no knowledge about the content of the documents. In IND-CKA1, the index structure for each document contains an equal number of keywords, while in IND-CKA2 this requirement is not imposed. These models offer robust security for the secure indexes, but they do not take into account the trapdoors.

In [10], the authors extend the application of the IND-CKA1 and IND-CKA2 security models to the trapdoor values in a symmetric searchable encryption scheme. In the less stronger model, IND-CKA1, the unauthorized user does not take into account the results of previous search queries or previous trapdoors when challenging a current search query. In the more strong security model, IND-CKA2, this limitation is removed for the malicious user.

4.3.2.2 Security Models for PKSE

Adaptive chosen keyword attacks

Similarly, in their work [2], the authors propose the security model PK-CKA2 (Chosen Keyword Attack for Public-Key encryption with keyword search—the stronger version). This security model provides protection against adaptive chosen keyword attacks, ensuring that unauthorized users gain no knowledge about the keywords when analyzing the ciphertext. In the PK-CKA2 model, unauthorized users are able to compute the trapdoor value tw for any keyword w, except for the challenge

keyword. The unauthorized user, acting as an adversary, proceeds as follows: the challenger receives two challenges for the keywords w_1 and w_2 from the adversary, for which the challenger generates a random bit $b \in \{0, 1\}$. Subsequently, the challenger encrypts b as w_b and sends the encrypted value to the adversary. Finally, the adversary attempts to establish the value of b, while also being able to determine the trapdoor values for other keywords. In the PK-CKA2 model, the adversary is unable to determine whether the ciphertext w_b corresponds to w_1 or w_2. The security model is described below [2]:

1. The challenger $\mathcal{C}$ generates a pair of keys $(key_{public}, key_{secret})$ by executing the `KeyGeneration` algorithm. The challenger $\mathcal{C}$ keeps key_{secret} confidential and discloses key_{public}.
2. The adversary $\mathcal{A}$ may select a keyword $kw \in \{0, 1\}$ and adaptively request the corresponding trapdoor value t_{kw} from the challenger $\mathcal{C}$.
3. The adversary $\mathcal{A}$ can make an arbitrary number of requests before issuing a challenge. For the challenge, $\mathcal{A}$ sends two distinct words $w_1, w_2 \in \{0, 1\}$ from the word dictionary to the challenger $\mathcal{C}$. The condition is that $\mathcal{A}$ must not have previously requested the trapdoor values t_{w_1} and t_{w_2}. The challenger $\mathcal{C}$ randomly selects a word $w' \in \{w_1, w_2\}$ and encrypts it to get $c = \texttt{PEKS}(w', key_{public})$, where the `PEKS` algorithm is described in Sect. 4.2.2.2.
4. After the challenge, the adversary $\mathcal{A}$ may continue to request trapdoor values, with the same restriction that it may not request the trapdoors t_{w_1} and t_{w_2}.
5. At a time of its choosing, the adversary outputs a value $c' \in \{0, 1\}$. The adversary $\mathcal{A}$ wins the game if $c = c'$.

In this scenario, the adversary $\mathcal{A}$ is considered victorious if it accurately determines whether the encryption it receives corresponds to w_1 or w_2. To quantify the advantage of the adversary $\mathcal{A}$, we define it as follows (λ denotes the security parameter):

$$Adv_{\mathcal{A}}(\lambda) = Pr[c = c'] - \frac{1}{2}. \tag{4.3}$$

Definition 4.3 If every polynomial-time adversary $\mathcal{A}$ has only a negligible advantage in the aforementioned game, then the PEKS scheme is considered **secure against adaptive chosen keyword attacks**.

Off-line Keyword Guessing Attack

In the context of searchable encryption schemes, common threats include the Keyword Guessing Attack (KGA) and the File Injection Attack (FIA). The KGA was first introduced in the work [22] and has since been applied to various PEKS schemes.

There are two types of KGAs: one is initiated by an external attacker, an entity unrelated to the owner of the server or provider, which can monitor the public channel between the server and the receiver. In [23], a secure channel for PEKS schemes was proposed for the first time. In this setup, an adversary can obtain encrypted keywords when the trapdoor is transmitted via the public channel, but cannot execute the

search/test algorithm. The other type of KGA is initiated by an internal adversary, typically associated with a malicious server. This adversary can obtain encrypted keywords from anyone and, given the malicious server's access to trapdoor information, can run the search/test algorithm to discover the correlation between an encrypted keyword and the trapdoor.

Let's denote the attacker as $\mathcal{A}$. The KGA for PEKS proceeds as follows [24]:

1. The challenger generates the public and private keys, sends the public key to the attacker, and keeps the private key secret.
2. The adversary $\mathcal{A}$ may select a keyword $kw \in \{0, 1\}^*$ and adaptively request the trapdoor value t_{kw} of the keyword kw from the challenger $\mathcal{C}$.
3. After requesting various trapdoor values, at a time of its choosing, the adversary $\mathcal{A}$ selects two keywords w_0 and w_1 and sends them to the challenger $\mathcal{C}$. Then, $\mathcal{C}$ randomly chooses a value $b \in \{0, 1\}$ and computes the challenge trapdoor $t_{w_b} = \texttt{Trapdoor}(key_{secret}, w_b)$ (where `Trapdoor` algorithm is presented in Sect. 4.2.2.2), which is then sent to $\mathcal{A}$.
4. After the challenge, the adversary $\mathcal{A}$ may further request trapdoor values, with the following restriction: it may not request the trapdoors t_{w_0} and t_{w_1}.
5. At a time of its choosing, the adversary outputs a value $b' \in \{0, 1\}$. The adversary $\mathcal{A}$ wins the game if $b = b'$.

The advantage of the adversary $\mathcal{A}$ is defined as follows, where λ represents the security parameter:

$$Adv_{\mathcal{A}}(\lambda) = Pr[b = b'] - \frac{1}{2}. \tag{4.4}$$

Definition 4.4 A PEKS scheme is considered **secure against an adaptive keyword guessing attack** if any polynomial-time attacker $\mathscr{A}$ has an advantage that is a negligible function in the aforementioned game.

File Injection Attack

The File Injection Attack (FIA) for PEKS was first analyzed in the work [25]. In this type of attack, a malicious server can gain information about queries by injecting a file into the data user's system. An FIA can leak the access pattern, making it easier for the adversary to recover a significant amount of data.

4.4 Classification of Search Operations

Before analyzing the types of elements based on which the search is made, first, we should make a remark related to the *number* of keywords accepted by the search algorithm. Therefore, there are two categories in this regard:

- *Single Search* (or, often called *Single-Keyword Search*): the search algorithm accepts only one keyword;
- *Multi-Search* (or, often called *Multi-Keyword Search*): the search algorithm accepts multiple keywords. Here, based on the relation between the keywords, there are three main categories: *Exact Keyword SE* (when there is no relation between the keywords and are returned all documents that contain at least one keyword), *Conjunctive Keyword SE* (when the keywords are linked by the AND operator and are returned all documents that contain all keywords), and *Boolean Keyword Search* (when the keywords are linked by any Boolean operator—AND, NOT, OR). However, the classification in one of these categories is not that strict, for example, a Conjunctive Keyword Search can be actually categorized as Boolean Keyword Search.

In the research literature, different types of search operations have been proposed. In the following section, we will categorize (SE) schemes based on the search operation, as outlined in the study by Pham et al. [6].

4.4.1 Search Based on Keywords

Sequential scan

The sequential scan is based on keywords that describe documents. The initial work in searchable encryption was presented in [1], where each document's descriptive keywords w_i are encrypted twice using independent methods. Firstly, an encryption scheme E is used to encrypt the keyword w_i, resulting in an output $c_i = E(w_i)$ represented in n bits. This output is divided into two parts: the right side r_i represented in m bits, where $0 < m \leq n$, and the left side l_i represented in the remaining $n - m$ bits. Subsequently, values $s_1, \ldots, s_i$, with s_i represented in $n - m$ bits, are generated for use in a stream cipher (as defined in [1]) that encrypts l_i. The second encryption requires the value t_i calculated as $t_i =< s_i, F_{k_i}(s_i) >$, where $k_i = f_{k'}(l_i)$. The encrypted value C_i resulting from the second encryption is $C_i = c_i \oplus t_i$, where $\oplus$ denotes the XOR operator. The server receives the value C_i. To search for documents containing a specific keyword w, the values $c = E(w)$ and $k = f_{k'}(L)$ are computed in the first step, and then the search query $< c, k >$ is submitted to the server. In the search algorithm, the server checks if $C \oplus c$ can be represented as $< s, f_k(s) >$ for some s. If the server finds a value C that satisfies the previous condition, the search query's output is the encrypted document that C describes. For decryption, a pseudorandom generator produces the value S_i, which is XORed with the first $(n - m)$ bits of c_i. Consequently, the value l_i used to compute k_i is obtained. The plain keyword w_i is derived from k_i. The scheme proposed in [1] maintains the privacy of the plain data and the query keyword. The time complexity is linear in the number of documents, leading to the technique's name, *sequential scan*. However, the scheme

is susceptible to statistical attacks, as the frequency of a query keyword's appearance in a document can be calculated.

The work in [2] presents another searchable encryption scheme, marking the first instance of public-key encryption with keyword search. This scheme is based on *Identity-based Encryption* (IBE) and consists of three phases. An encryption scheme E is used, where the scheme's public key is the receiver's public identity, employed to encrypt the documents and keywords. The second phase involves computing the trapdoor value t_w for a query keyword w', which is then sent to the server. Subsequently, the server executes the search process and identifies the documents with keywords matching the trapdoor value. The mathematical framework for this PEKS involves two groups G_1, G_2 of prime order p. The scheme is secure against the PK-CKA2 attack.

In [26], the authors utilize Cocks IBE Scheme [27] for their proposed searchable encryption scheme. The hardness problem is quadratic residuosity. During the search process, the server computes $4k$ Jacobi symbols [28], where k is the scheme's security parameter. The time complexity is linear in the number of ciphertexts.

The paper [29] introduces another searchable encryption scheme based on k-resilient identity-based encryption. Although the scheme proposed in this work is foundational, it forms the basis for two other searchable encryption schemes that eliminate the need for a secure channel for trapdoors. A sequential scan SE scheme supporting document updates is proposed in [30]. Here, the primary encryption scheme comprises two basic encryption schemes: one interactive (where a sub-algorithm interacts with the server during the key generation phase to create a meta-data structure, and the search algorithm also interacts with the server) and one non-interactive (where interaction with the server is removed) [30].

Index-based

The authors of [17, 19] propose searchable encryption schemes that are based on *secure indexes associated with the documents*, which enhance efficiency. In [17], the keywords describing the documents undergo two rounds of processing by pseudo-random functions. The search algorithm employs a Bloom filter [31], which generally checks if an element belongs to a set. In this specific case, it checks if a keyword belongs to a set describing a document. The server performs the search on secure indexes, not on the entire ciphertext as in the system proposed in [17], leading to improved efficiency. Consequently, the time complexity is linear in the number of documents.

In [10], the authors utilize a different type of secure indexes—these are *related to the keywords* describing the documents, not the documents themselves as in other examples (for differences between types of indexes, see Fig. 4.3). In this scheme, the keyword w is associated with the identifiers of the documents it describes. This approach is more efficient, achieving time complexity linear in the number of documents containing the searched keyword. However, updating the set of keywords in this approach is quite challenging.

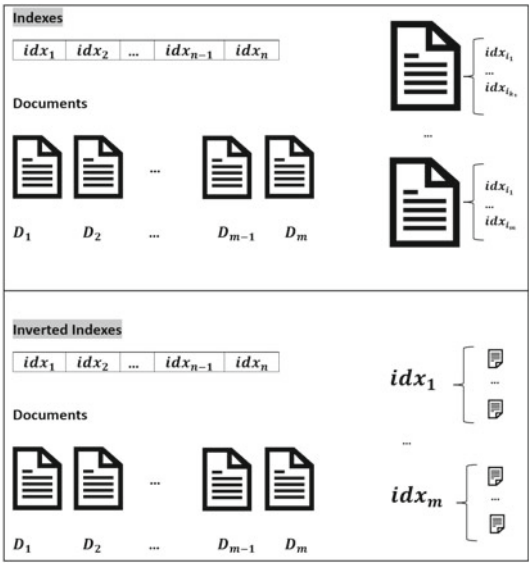

Fig. 4.3 Difference between regular indexes and inverted indexes

The paper [32] extends the work of [1] by representing keywords through hashed indexes. Additionally, a randomized homomorphic scheme based on XOR is included at an additional level for the search query. The authors of [33] propose cloud indexes constructed for different types of data in the cloud.

Another paper, [34], presents a scheme where secure indexes are used. This work extends the scheme presented in [10]. However, it was quickly followed by [35], as the initial version of the scheme leaked information about the trapdoor values. The issue was resolved in the second version, which also supports document update operations.

4.4.2 Search Based on Regular Expressions

The research papers [36, 37] introduce the RESeED framework, which is based on regular expressions. In this framework, the search process involves looking for a regular expression within encrypted documents. The system comprises two components: the Column Store and the Order Store. The Column Store, structured as an inverted index, maintains the documents and their corresponding keywords. On the other hand, the Order Store retains the same keywords but in a hashed form. The search process employs a Nondeterministic Finite Automaton (NFA), which is divided into sub-NFAs. These sub-NFAs are used to find matches between the query keywords and the keywords from the Column Store. Subsequently, the Order Store is utilized to verify if the keywords appear in the same sequence as in the regular expression, for the documents obtained in the previous step.

4.4.3 Semantic Search

Searchable encryption schemes that allow for more flexible searches are often necessary, as there are instances where users require a search process that looks for close matches to keywords, rather than exact matches. This section presents such searchable encryption schemes.

Fuzzy search

Fuzzy-based approaches are used when the search algorithm is designed to search for close matches of keywords. An example of a scheme that uses a fuzzy-based search is [38], where user's typos are accepted as query keywords. In [39], the dissimilarity between two strings is measured by a function called *edit distance* (*ed*). This dissimilarity is defined as the number of operations needed to transform the first string into the second string, with the permissible operations being replacement, insertion, and deletion. The elements required for the search process include the set D of n documents, the set w of n unique keywords associated with each document, the value d of the edit distance, and the trapdoor value (s, k), where $k \leq d$. Given the query keyword kw, if $kw \in w$, then a fuzzy system will return the set of documents that could possibly contain kw. Otherwise, it will return the document for which $ed(kw, w_i) < k$, $w_i \in w$. Another example of a fuzzy technique is presented in [40], where for each keyword kw, a set of variations of kw is generated, with $ed(kw, w_i) < d$, where w_i is the variation and d is a predefined edit distance. The sets of variations are stored on the same server where the index structure is stored. Before initiating the search process, the variations of a query keyword are computed and all sent to the server. The server then searches for matches within the variations sets. As this technique is space-intensive, a wildcard symbol $*$ was introduced to represent the fuzzy character or to omit a character. This improvement leads to better space complexity, and therefore, a better search time. Other research studies in fuzzy matching include [41], which uses a dictionary-based fuzzy search, and [42], which is based on a fuzzy keyword tree.

Stemming search

While the fuzzy approach offers more flexibility than exact matching, it sometimes fails to cover semantic similarity due to a large edit distance. For instance, the words "student" and "studying" have a large edit distance, which a fuzzy system might not cover. This issue can be addressed by a variant of SE schemes known as semantic SE. The central idea behind semantic schemes is that semantically similar words share the same root, referred to as the *stem word*. The following techniques can be used to compute the stem word:

- *Affix removing*, which involves removing the prefixes and suffixes of a word to obtain its root. Algorithms that perform this operation include J.B. Lovins' algorithm [43] and Porter's algorithm [43]. To utilize this approach, searchable encryp-

tion schemes should incorporate computational linguistics or natural language processing techniques, which entail a significant computational cost.
- *Statistical stemming*, which involves using an n-gram stemming algorithm to compute the minimum sequence of characters that frequently appear in a given text [44].
- A hybrid technique that employs both affix removing and statistical stemming.

Ontological

Searchable encryption schemes can yield even more flexible and accurate results through ontological SE schemes, which return keywords that fall within the semantic field of the query keyword. Several techniques can be used to compute the semantic field of a word:

- *Semantic relationship*, which calculates the similarity of two words based on the semantic relationship between words in the co-occurrence of words. To compute the co-occurrence factor for the elements of a given set, the authors of [45] used the following formula from data mining:

$$F(s_1, s_2) \equiv \log_2 \frac{P(s_1, s_2)}{p(s_1)p(s_2)} \tag{4.5}$$

  where s_1 and s_2 are two strings, $P(s_1, s_2)$ represents the probability that the strings occur together, and $p(s_1)$ and $p(s_2)$ represent the probability that the string occurs independently in the set.
  Another algorithm that computes the similarity between two words is the *Cosine similarity* [8, 45]. To use this algorithm to compute the resemblance between the query keywords and the encrypted documents, the documents and the corresponding keywords should be represented as vectors.
- *Inverted indexes*, which is the opposite of the initial technique: for each unique keyword, the set of document identifiers to which it belongs is stored. Moreover, the document identifiers are given a ranking score (a value between [0, 1] when the score is normalized) that expresses the relevance of the keyword in the document [13, 15, 45].
- *The ranking function*, which is similar to the inverted indexes. When the query search is initiated, the server provides the documents that match the query search in terms of relevance. This technique uses a ranking function that outputs the degree of relevance of the query keyword in the current document. A widely used ranking function is TFxIDF [14, 45], which stands for term frequency (the relevance of the keyword for the document) versus inverted document frequency (the relevance of the keyword for all documents).

Numerous ontological SE schemes have been proposed in the research literature: [12, 45] use inverted indexes and a semantic relationship library for encrypted metadata about each document; [14] introduces Synonym-based Keyword Search (SBKE), Wikipedia-based Keyword Search (WBKS) and a combination of these, WBSKS,

which uses TFxIDF and vector representation; [46] extends the retrieval technique Okapi BM25 using the TFxIDF function to find relevance between query search and the set of the documents; [47] uses the **WordNet** dictionary [32] to add the synonyms of the keyword query to the search query.

Figure 4.4 presents a summary of SE types and characteristics from different perspectives.

4.5 Advancements in Searchable Encryption

4.5.1 Dynamic Searchable Encryption

In a regular SSE scheme, the data owner encrypts their data and uploads it to the server. The data owner also generates search tokens for each keyword in the data set. When a search is performed, the server uses the corresponding search token to identify and return the relevant encrypted documents.

However, this regular SSE scheme does not support updates to the data set. If the data owner wants to add, delete, or modify a document in the encrypted data set, they would need to re-encrypt the entire data set, which is not practical for large data sets or for data sets that are updated frequently.

An extended form of symmetric searchable encryption is DSE. This variant lets the user to add or delete files directly on the server, eliminating the need to retrieve them first. This is achieved by using specific tokens (or trapdoors) designated for the addition and removal processes. In addition to the previously mentioned algorithms, dynamic SSE schemes incorporate the following algorithms [48]:

- `AddToken`$(key_{secret}, D) \rightarrow (T_{add}, C_D)$: This algorithm takes the secret key key_{secret} and the document D to be added as input, and outputs the adding token T_{add} and the encrypted document C_D.
- `DeleteToken`$(key_{secret}, D) \rightarrow T_{del}$: This algorithm takes the secret key key_{secret} and the document D to be deleted as input, and outputs the deletion token T_{del}.
- `AddDocument`$(I, C, T_{add}, C_D) \rightarrow (I', C')$: This algorithm takes the encrypted index I, the set C of encrypted documents, the adding token, and the encrypted document C_D as input, and outputs the updated encrypted index I' and the updated set of encrypted documents C'.
- `DeleteDocument`$(I, C, T_{del}) \rightarrow (I', C')$: This algorithm takes the encrypted index I, the set C of encrypted documents, and the deletion token T_{del} as input, and outputs the updated encrypted index I' and the updated set of encrypted documents C' after the deletion of the specified document.

The correctness of a dynamic SSE encryption scheme is based on the condition that for any secret key key_{secret} produced by `KeyGeneration`, for any function f, and for any (I, C) resulting from `Encryption`(key_{secret}, D), any sequence of

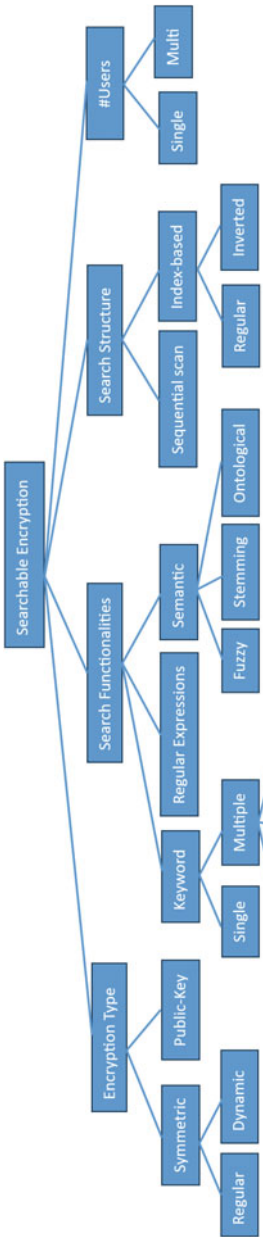

Fig. 4.4 Summary of SE schemes

`AddDocument`, `DeleteDocument`, and `Search` algorithms applied on I should always return the correct sequence of indices.

In [48], the authors define the CKA2 security for dynamic SSE schemes as follows:

Definition 4.5 Consider $\varepsilon = (\texttt{KeyGeneration}, \texttt{Encryption}, \texttt{Trapdoor}, \texttt{AddToken}, \texttt{DeleteToken}, \texttt{Search}, \texttt{AddDocument}, \texttt{DeleteDocument})$ as a dynamic symmetric SE scheme that employs indexes. Let $\texttt{Real}\mathcal{A}(k)$ and $\texttt{Ideal}\mathcal{A}, \mathcal{S}(k)$ be two probabilistic experiments defined as follows, where $\mathcal{A}$ denotes a stateful adversary, $\mathcal{S}$ denotes a stateful simulator, and $\mathcal{L}_i$ denotes a stateful leakage algorithm, $i = \overline{1,4}$:

`Real`$\mathcal{A}(\lambda)$: The challenger $\mathcal{C}$ uses the `KeyGeneration` algorithm to obtain the private key key_{secret}. The adversary $\mathcal{A}$ sends D to the challenger $\mathcal{C}$ and receives (I, C) from $\texttt{Encryption}(key_{secret}, D) \rightarrow (I, D)$. The adversary $\mathcal{A}$ requests w, D_1, D_2. For any query q, $\mathcal{A}$ receives from $\mathcal{C}$ the values T_w, (T_{add}, C_{D_1}) or T_{del} for D_2, obtained from the `Trapdoor`, `AddToken`, `DeleteToken` algorithms with the appropriate parameters. After a polynomial number of adaptive queries, A outputs the bit b, which is the result of the experiment.

`Ideal`$_{\mathcal{A},\mathcal{S}}(k)$: The adversary $\mathcal{A}$ sends D to the simulator $\mathcal{S}$, which uses $\mathcal{L}_1(D)$ to obtain (I, C) that is given to $\mathcal{A}$. The adversary A makes queries of form $q \in w$, D_1, D_2, where for any q the simulator $\mathcal{S}$ sends to $\mathcal{A}$ one of the following outputs $\mathcal{L}_2(f, w)$, $\mathcal{L}_3(f, D_1)$ or $\mathcal{L}_3(f, D_2)$. The simulator outputs the token T and the encrypted text C (if the operation is for adding a document), and A outputs the bit b, which is the result of the experiment.

Given these settings, if for any PPT adversary $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ with the property below, then ε is called $(\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4)$—**secure against adaptive dynamic CKA**:

$$\left| Pr[\texttt{Real}\mathcal{A}(k) = 1] - Pr[\texttt{Ideal}\mathcal{A}, \mathcal{S}(k) = 1] \right| \leq negl(k). \tag{4.6}$$

Other types of security definitions involve dynamic SSE schemes. In [49], the authors informally introduced the concepts of *forward and backward privacy*, while in [50], the authors formally described these concepts.

Intuitively, *forward privacy* implies that the *update* process (referring to both `AddDocument` and `DeleteDocument` algorithms) does not leak information about the keywords used in the (Document, keyword) pair input for the update algorithm.

Definition 4.6 Consider ε as a dynamic SSE scheme and $\mathcal{L}^{Update}$ as the update leakage function. We say that ε is **forward-private** if and only if $\mathcal{L}^{Update}$ can be expressed as follows:

$$\mathcal{L}^{Update}(op, in) = \mathcal{L}'(op, (D_i, \mu_i)) \tag{4.7}$$

where op represents the operation (add or delete), (D_i), μ_i is the set that includes the document D_i updated with a number μ_i of keywords, and $\mathcal{L}'$ represents a stateless leakage function.

If the update process accepts only one pair (Document, keyword), then the condition 4.7 simplifies to $\mathcal{L}^{Update}(op, kw, D) = \mathcal{L}'(op, D)$.

Intuitively, *backward privacy* implies that the search process based on a keyword kw does not leak information about the document D that contains kw when the search is conducted between the operations of adding the pair (D, kw) and deleting the pair (D, kw). There are three types of backward privacy, as follows:

- **Backward privacy with insertion pattern** implies that the documents that *currently* contain the keyword kw are leaked during the insertion process. Additionally, the number of updates applied to the keyword kw is leaked.
- **Backward privacy with update pattern** implies that the documents that *currently* contain the keyword kw are leaked during the insertion process and each time kw is updated.
- **Weak backward privacy** implies that the documents that *currently* contain the keyword kw are leaked at the time of insertion, at the time of updating kw, and from the information about which insertion was cancelled by which deletion.

To formalize these types of backward privacy, the authors in [50] defined additional leakage functions, which we describe below. The leakage function $\texttt{TimeDocs}(kw)$ is the set of documents D that contain kw (excluding the documents that were deleted) and the timestamp t at which these were **added** to the server. Given a query set Q, then $\texttt{TimeDocs}(kw)$ can be defined as:

$$\texttt{TimeDocs}(kw) = \{(t, D) \mid (t, \texttt{add}, (kw, D)) \in Q \quad \text{and} \quad \forall t', (t', \texttt{delete}, (kw, D)) \notin Q\} \tag{4.8}$$

A version that provides less information than $\texttt{TimeDocs}$ is $\texttt{Docs}(kw) = D \mid \exists t \quad \text{such that} \quad (t, D) \in \texttt{TimeDocs}(kw)$. The set that retains the timestamp t at which the keyword kw was **updated** is defined as follows:

$$\texttt{UpdateOps}(kw) = \left\{t \mid (t, \texttt{op}, (kw, D)) \in Q, \texttt{op} \in \{\texttt{add}, \texttt{delete}\}\right\} \tag{4.9}$$

The final additional set includes the timestamp $t^{\texttt{add}}$ for each delete operation for keyword kw paired with the timestamp $t^{\texttt{delete}}$ that provides the corresponding time for the insert operation which the delete operation cancels.

$$\begin{aligned} &\mathcal{L}^{Update}(op, kw, D) = \mathcal{L}'(op) \\ &\mathcal{L}^{Search}(kw) = \mathcal{L}''(\texttt{TimeDocs}(kw), a_{kw}) \end{aligned} \tag{4.10}$$

where op represents the operation (add or delete), a_{kw} represents the number of entries for the keyword kw (even deletions are included here) and $\mathcal{L}'$ and $\mathcal{L}''$ are stateless leakage functions.

Definition 4.7 Let ε be a dynamic SSE scheme, $\mathcal{L}^{Update}$ be the update leakage function and $\mathcal{L}^{Search}$ be the search leakage function. We say that ε is **backward-private with update pattern** if and only if $\mathcal{L}^{Update}$ and $\mathcal{L}^{Search}$ can be expressed as follows:

$$\begin{aligned}\mathcal{L}^{Update}(op, kw, D) &= \mathcal{L}'(op, kw)\\ \mathcal{L}^{Search}(kw) &= \mathcal{L}''(\texttt{TimeDocs}(kw), \texttt{UpdateOps}(kw))\end{aligned} \tag{4.11}$$

where op represents the operation (add or delete) and $\mathcal{L}'$ and $\mathcal{L}''$ are stateless leakage functions.

Definition 4.8 Let ε be a dynamic SSE scheme, $\mathcal{L}^{Update}$ be the update leakage function and $\mathcal{L}^{Search}$ be the search leakage function. We say that ε is **weak backward private** if and only if $\mathcal{L}^{Update}$ and $\mathcal{L}^{Search}$ can be expressed as follows:

$$\begin{aligned}\mathcal{L}^{Update}(op, kw, D) &= \mathcal{L}'(op, kw)\\ \mathcal{L}^{Search}(kw) &= \mathcal{L}''(\texttt{TimeDocs}(kw), \texttt{DeleteHist}(kw))\end{aligned} \tag{4.12}$$

where op represents the operation (add or delete) and $\mathcal{L}'$ and $\mathcal{L}''$ are stateless leakage functions.

The relationship among the three definitions is that insertion pattern backward privacy implies update pattern backward privacy, which in turn implies weak backward privacy. From Definition 4.8, we can observe that insertion pattern backward privacy should also be forward private. Therefore, when a dynamic SSE scheme ε simultaneously satisfies forward privacy and weak backward privacy, the leakage generated from the update queries does not depend on the keyword for which the update process is executed, nor on the index of the updated document.

DSE schemes have evolved significantly over time, addressing several challenges that were prevalent in earlier designs. Initially, DSE schemes were inefficient, often leading to high computational costs during updates and searches, particularly for large databases. Modern DSE schemes have made substantial improvements in enhancing efficiency, with many of them offering sub-linear or constant time complexity for these operations. The lack of forward and backward privacy was another notable drawback in older DSE schemes. Recent schemes, however, typically ensure both, protecting the system against revealing information about new additions or deleted documents in relation to past or future queries, respectively. Older DSE schemes also were exposed to significant data leakage, including search and access pattern leakage. Contemporary schemes have made important progress in minimizing such leakages, with some even providing "leakage-abuse" free operation. Moreover, while early DSE schemes were secure primarily against passive adversaries, modern schemes have been designed to resist to more powerful, active adversaries. Scalability was another challenge for older DSE schemes, which often struggled with large-scale databases. This issue has been addressed in recent designs, which are built to handle large-scale, real-world databases. Lastly, the support for complex queries has been more explored in modern DSE schemes, moving beyond simple keyword searches to accommodate Boolean queries, range queries, and certain forms of conjunctive and disjunctive queries.

Janus

One of the most important works in DSE is presented in [50]. In this work, the authors introduce the `Janus` SE scheme. One of its advantages stays in the fact that it is designed to provide both forward and backward privacy, a feature that is important in ensuring the confidentiality of user queries and the integrity of encrypted data. As we have seen, forward privacy ensures that an adversary cannot learn about new documents added to the database, while backward privacy protects against information leakage about deleted documents. The Janus scheme achieves this by using constrained cryptographic primitives, which are designed to limit the capabilities of potential adversaries. This approach not only enhances the security of the system but also improves its efficiency, making it a practical solution for real-world applications. The introduction of Janus has had a profound impact on the field, setting a new standard for privacy in searchable encryption. It has served as a foundational model for several subsequent schemes, providing a robust structure for achieving both forward and backward privacy in searchable encryption.

For instance, Janus++ [51] is an improved version of the Janus framework that not only maintains the privacy features of the original model but also enhances its security and efficiency. The Janus framework has also been adapted for use with Intel's Software Guard Extensions (SGX) to provide practical performance while ensuring privacy [50].

Moreover, the Janus framework's use of constrained cryptographic primitives has inspired new constructions of searchable encryption schemes that achieve forward and backward privacy. These subsequent schemes have further expanded the applicability and effectiveness of the Janus framework, demonstrating its importance and influence in the ongoing development of searchable encryption.

Further, we will present the algorithms of the Janus scheme [50], but before introducing this DSE, we first need a general background for textitincremental puncturable encryption scheme (IPE), which is the basis for Janus. A puncturable encryption (PE) scheme is a cryptographic system that allows for the encryption of data in such a way that the decryption key can be "punctured" for specific ciphertexts. This means that once a key has been punctured for a particular ciphertext, it can no longer be used to decrypt that ciphertext, but it can still decrypt other ciphertexts for which it has not been punctured. This technique was introduced in [52] and it was initially used to achieve forward secure encryption for informational systems used for storing or forwarding messages (emails, SMSs, etc.). The "incremental" aspect of IPE refers to the ability to puncture the key for additional ciphertexts over time. This is a powerful feature because it allows for fine-grained control over access to encrypted data. For example, in a searchable encryption context, an IPE scheme can be used to ensure that once a search has been performed, the server cannot use the same key to learn anything about the same search query in the future, providing a form of forward privacy. In order to use a puncturable encryption scheme (often known as PPKE), the message space $\mathcal{M}$ should be defined together with the tag space $\mathcal{T}$. A PPKE scheme consists of four algorithms (`KeyGeneration`, `Encryption`, `Puncture`, `Decryption`) that have the following goals [52]:

- `KeyGeneration`(1^{λ}): generates a pair (PK, SK_0) consisting of public key and an *initial* private key.
- `Encryption`(PK, M, t): for a tag $t \in \mathcal{T}$ having attached a message $m \in \mathcal{M}$, it computes the encrypted message CT.
- `Puncture`(SK_i, t): generates a new private key SK_{i+1} from the current SK_i. This new private key can decrypt the same ciphertexts as the current secret key, except for those constructed based on the tag t.
- `Decryption`(SK_i, CT, t): decrypts the encrypted message CT.

The property of correctness in this context is achieved if a plaintext message M, encrypted with a specific tag t, can be successfully decrypted back to M using a secret key that has been punctured on any set of tags, provided that this set does not include the tag t.

Now, coming back to Janus, firstly, the authors propose in [50] an improvement for the punctural encryption scheme proposed in [52]. The `Puncture` algorithm becomes

$$\texttt{Puncture}(SK_n, t) = (sk'_0, sk_1, \ldots, sk_{n+1}),$$

with $(sk'_0, sk_{n+1}) = \texttt{IncPuncture}(sk_0, t)$ [50]. The change here is that the secret key is fractioned in a constant length, becoming the basis for the `Puncture` algorithm. There are involved two forward-secure SSE schemes involved: Σ_{add} (insertion instances) and Σ_{del} (deletion instances). The first one is used to store indices that are newly added and which are encrypted with the PE scheme, while the second one stores the punctured key elements, considering that for each keyword there is a different encryption key. Initially, the client should host on the local environment a table sk_0 consisting of the initial key share ($sk_0[w]$) of the PE for every keyword w. The Janus algorithms are as follows [50]:

- `Add`(w, ind): this algorithm adds a new file ind described by the keyword w, by encrypting the pair with the PE scheme for which the secret key $sk_0[w]$ and the tag is generated by a pseudorandom function $F(w, ind)$. The ciphertext is inserted in the SSE instance Σ_{add} as a new entry that matches w.
- `Delete`(w, ind): this algorithm deletes a new file ind described by the keyword w, the tag $t = F(w, ind)$ and the secret key $sk_0[w]$ corresponding to w is punctured to the tag, obtaining sk_t. Then, $sk_0[w]$ is updated with sk_t. The deleted entry (w, sk_t) is inserted in the SSE instance Σ_{del}.
- `Search`(w): To search for a keyword w, both instances of SSE Σ_{add} and Σ_{del} are triggered by the client. The server, that discovers the results, can access the encrypted *inserted* indices, which are sent to the client, but the client can decrypt the ciphertext only with non-punctured secret keys, meaning these correspond to the entries that have not been deleted.

Using this approach, when the search process is triggered, each time a new secret key associated with the query word will be generated, therefore when a new document is inserted, this new key will be used. The authors prove that Janus ensures forward-privacy and weakly backward-privacy. Janus is the first non-interactive backward-

secure SSE, meaning that provides backward security without requiring interaction between the client and the server during the update process.

Janus has represented an important achievement in the DSE domain. However, study [51] has shown that in practice Janus is quite inefficient. In this regard, in [51] an improvement is proposed, leading to the scheme Janus++. The important observation that the authors made is that PE is a type of PKI, meaning it uses a public key, while DSE is a type of SSE, meaning that at first sight, a public key would not be necessary. Indeed, the authors propose in [51] a symmetric PE (SPE), eliminating the need for the public key. In contrast to Janus, the approach from [51] limits the number of permissible deletions d between two consecutive search queries. However, depending on the workload, this restriction might be modified for various keywords or even the same phrase during runtime. This is due to the fact that after each search, both the newly inserted index entries and the punctured keys for a specific term are produced from a new master secret key. Previous results are cached by the server, and tokens created by a new master key query updated entries, removing the need to re-encrypt any entries. Since deletions are less often than searches, this is not a serious worry in practice. Furthermore, the instantiation from [51] allows batch deletion by default, allowing for multiple deletions in each of the total d punctures. This feature has the potential to increase the number of allowed removals.

The SPE has the following form $SPE = (\texttt{KeyGeneration}, \texttt{Encryption}, \texttt{IncrementalPuncturing}, \texttt{Decryption})$. Janus++ is constructed based on this SPE, more forward-private DSE schemes $\Sigma_{op} = (\Sigma_{op}.\texttt{Setup}, \Sigma_{op}.\texttt{Search}, \Sigma_{op}.\texttt{Update})$, $op \in \{add, del\}$, and a pseudo-random function F. The algorithms work as follows, as they are presented in [51] (below, we use the same notations as in [51]):

- $\texttt{Setup}(1^{\lambda})$: based on the security parameter λ, the client generates two values $K_s, K_t \xleftarrow{\$} \{1, 2\}^{\lambda}$. Based on the `Setup` algorithms of SSEs Σ_{add} and Σ_{del} the following tuples are generated: $(EDB_{add}, K_{add}, \sigma_{add})$ and $(EDB_{del}, K_{del}, \sigma_{del})$. Further, MSK (it stores the encryption key for every keyword), PSK (it stores the local key shares from which keys will be punctured), DEL (retains the allowed number of deletions for each keyword) SC (stores the number of queries for each keyword), EDB_{cache} (stores the most recent results) are initialized as empty sets. Therefore, the output from the client is $K_{\Sigma} = (K_{add}, K_{del}, K_s, K_t)$, $(EDB_{add}, EDB_{del}, EDB_{cache})$, $\sigma = (\sigma_{add}, \sigma_{del}, MSK, PSK, DEL, SC)$.
- $Search(K_{Sigma}, w, \sigma; EDB)$: the input of the client is K_{Sigma}, w, σ, which will be used to sear for a keyword w. In order to do this, the local key share msk' is retrieved $PSK[w]$, together with the number of times the keyword w was searched $SC[w]$. The client sends to the server msk' and the token value $tkn = F(K_s, w)$ that will be used to get the result of the last search query from the cache. Based on these values, the client runs the `Search` algorithm for both $\sigma_{add}, \sigma_{del}$ based on the input $w||i$. The server gets the encrypted indices and the updated punctured key shares after the most recent search on w in the search protocol. The server

may get indices that have not yet been punctured by merging these punctured key shares with msk'. Finally, the server delivers the search result, which includes the freshly recovered and cached indices (but not the recently deleted ones). It's worth noting that the w encryption key must be updated after each search. This new key will be used to encrypt new items matching w before the next search.

- $Update(K_{Sigma}, op, (w, ind), \sigma; EDB)$: in this algorithm the op can be *add* or *del*. To *add* a new pair (w, ind), the client uses the master key $msk = MSK[w]$ to encrypt ind based on the tag $F_{K_t}(w, ind)$. Further, the pair with the encrypted value and the tag is added in EDB_{add} at location w, more precisely, at $w||i$. To *delete* the pair (w, ind), the client should use the local key share msk' retrieved from $PSK[w]$ based on which the client punctures the current key psk_t using msk' using the tag $t = F_{K_t}(w, ind)$. Then, the local key share $PSK[w]$ is updated at $PSK[w]$ and the pair (psk_t, t) is inserted in EDB_{del}.

Janus and Janus++ are the basis for many DSE schemes. Some subsequent works include [53], which assures forward and backward privacy with non-interactive search and update operations that do not depend on the random oracle model. An even more recent scheme is Janus-V [54], which achieves enhanced backward security—specifically, backward privacy with an update pattern. The proposed scheme leverages key-homomorphic pseudorandom functions and needs only a single roundtrip to retrieve the search result.

Besides Janus, there are other approaches in the literature. For example a recent DSE scheme [55], offers a document-level update mechanism. Unlike other schemes that base their updates on keywords, this scheme performs deletions using the document identifier, marking a significant change of approach. Initially, the authors propose a first scheme that has forward and backward privacy and support the operation of deletion, and then, they improve the first scheme, making it more efficient and secure against external attacks. Further, we describe the regular scheme, called $DBP\text{-}B$ (Document-based Backward Private DSSE) presented in [55]. Initially, the authors define a pseudorandom function $F : \{0,1\}^{\lambda} \times \{0,1\}^{*} \rightarrow \{0,1\}^{\lambda}$ and a keyed hash function $H : \{0,1\}^{*} \times \{0,1\}^{*} \rightarrow \{0,1\}^{2\lambda}$, and establish the structures that will be used: pairs of type $(key, value)$ are defined, where key is the unique identifier of the document, while $value$ is a list of keywords associated with the document. These pairs are stored in a map T_e. Another map is T_p that stores the most recent search results, and it is updated with each search operation. The algorithms work as follows:

- `Setup`: Let k_1 be the secret key for F, $key^{(w)}$ the secret key for a keywords w, T_c a map that stores keys for all keywords. These components are maintained by the DO through a *state* $\sigma = (k_1, T_c)$. Another two maps are generated by the DO, as follows: T_p stores the result of the last search query, and T_e stores the encrypted indexes. The last two components are contained by an *encrypted database* $EDB = (T_p, T_e)$, which is sent to the server.
- `Addition`: This algorithm is set to add a document. Let $f = (ind, DB(ind))$ be the document added by the DO, where ind is the identifier and $BD(ind)$ is the set of associated keywords. Firstly, the DO computes a token $\tau_a = (F_{k_1(ins),AddSet}$

which is sent to the server. The DO constructs the set $AddSet$ based on encrypted indexes as follows: using T_c, the DO extracts the private keys $key^{(w)}$ corresponding to each keyword $w \in DB(ind)$. The pair $\langle ind, 0 \rangle$ is then hashed using $e = H_1(F_{k_1}(ind), key^{(w)}$, which is inserted in $AddSet$.

- `Token`: This algorithm searches for documents based on keywords. Firstly, the DU generates randomly a string from $\{0, 1\}^{\lambda}$, extracts from T_c the key $key^{(w)}$ corresponding to the keyword, and updates $T_c[w]$ with the random string previously generated. This ensures that in every search, there is a new search key that will be used to generate the search token $\tau_s = (F_{k_1}(w), key^{(w)})$. This approach leads to forward privacy because the previous tokens cannot be used to search newly added documents.
- `Search`: Once received by the server, τ_s is parsed as $(\alpha, key^{(w)})$. For searching in the map T_p the value α, while for searching in the map T_e the key is used. The search process works as follows: a value called $mask = H_1(key, key^{(w)})$ is generated, then for each component $e \in ValueSet$, the plain text $\langle x, y \rangle$ containing $mask$ is recovered. From this pair, if $y = 0$, then x is an identifier for a document that matches the keyword. As such, x is added in Res, e is deleted from $ValueSet$, and the search is continued to non-empty $(key, value)$ pair from T_e. If $y! = 0$, then x is not an identifier for a document that matches the keyword, and the search continues with the next element from $ValueSet$. When T_p is queried, the last search result located at $T_p[\alpha]$ is stored into Res, and $T_p[\alpha]$ is updated with the new result. To be more efficient, for a found file, its identifier is removed from T_e, meaning that unnecessary testing is avoided in further search processes. The DU will receive the value Res.
- `Deletion`: This algorithm removes a file having the identifier ind. For this, the DO computes $F_{k_1}(ind)$ and the token τ_d and sends them to the server. To delete from T_e, the server removes the element at location $F_{k_1}(ind)$, while for deleting from T_p, this set is iterated and when ind is found, the entry is deleted.

The authors propose an extended version, DBP-E, that uses the notations and functions of DBP-B, while also introducing some additional notations. It will also use the encryption (Enc) and decryption (Dec) algorithms of a given symmetric cryptosystem. The construction of DBP-E shares similarities to that of DBP-B, although with some nuanced modifications, as follows:

- `Setup`: The client generates a pair of two private keys (k_1, k_2), where k_1 is generated as in the regular algorithms, and k_2 is the private key for the given symmetric cryptosystem.
- `Addition`: This is similar with the regular algorithm, the client encrypts the ind with the encryption algorithm of the given cryptosystem and then applies H_1 on the encrypted index.
- `Token`: This algorithm works as in the regular version.
- `Search`: The change here is that the server achieved encrypted indexes from T_e and T_p. So, the results sent to the client will be encrypted.

- `Deletion`: To delete a file, the process is similar to the regular process, with a slight change: the client sends to the server $Enc(k_2, ind)$ instead of ind. Then, the corresponding element is deleted from T_p.

The authors prove that their regular scheme is adaptively-secure in the random oracle model, and the improved version is $\mathcal{L}$-adaptively-secure with the leakage functions $\mathcal{L}$ defined as follows:

- $\mathcal{L}_{setup}(\lambda) = \emptyset$
- $\mathcal{L}_{addition}(ind, DB(ind)) = |DB(ind)|$
- $\mathcal{L}_{deletion}(ind) = ind$
- $\mathcal{L}_{search}(w) = (SP(w), TimeDB(w))$

Having these characteristics of DSE, studies have shown that a regular SSE can be transformed into a DSE. To transform a regular SSE scheme into a DSSE scheme, additional mechanisms are needed to handle updates to the encrypted data set. These mechanisms allow the data owner to add, delete, or modify documents in the encrypted data set without needing to re-encrypt the entire data set. The challenge in designing these mechanisms is to ensure that they do not compromise the security of the SSE scheme. For example, one approach to handle updates in a DSE scheme is to use a data structure called a binary tree, where each node in the tree corresponds to a document in the data set. When a document is added or deleted, only the corresponding node in the tree needs to be updated, which is much more efficient than re-encrypting the entire data set.

However, designing a DSE scheme is more complex than designing a regular SSE scheme, and there are many trade-offs to consider, such as the trade-off between search efficiency and security.

The first DSE scheme was proposed in [34]. This encryption scheme uses inverted indexes and it stands over the SSE scheme presented in [10], which is SSE. The SSE scheme from [10] works as follows: every file f is encrypted based on keywords from a set $w \in W$. For each of these keywords, it is constructed a list L_w that contains a number (this number is denoted with $\#f_w$) of nodes $(N_1, \ldots, N_{\#f_w})$ located at random positions in an array of searches A_s. Each node has the form $N_i =< id, addr_s(N_{i+1}) >$, where id represents the unique identifier of the file f that contains w and $addr_s(N)$ shows the location of N in A_s. The recommendation for A_s is to have the size of at least $|c|/8$ in order to not reveal statistical information about f. The rest of the cells that are not used are padded with random strings. Further, a search table T_s is constructed, by inserting the pointer to the head of the list L_w under the search key $F_{K_1}(w)$ (here, K_1 is the key to pseudo-random function F), for each keyword w. Then, using the secret key SKE is encrypted under the key generated as $G_{K_2}(w)$ (here, K_2 is the key to pseudo-random function G). For the search process, DU just sends the values of $F_{K_1}(w)$ and $G_{K_2}(w)$. Further, CS recovers the pointer to the head of the list L_w that contains w based on the T_s and $F_{K_1}(w)$. Then, using $G_{K_2}(w)$ the decryption can be made, and the identifiers of the files that contain w are revealed. This approach is not explicitly dynamic, because, based on the structure deletion or insertion, the pointers or nodes cannot be modified

due to encryption. In [34], the presented scheme is transformed into a DSE scheme as follows to overcome the limitations of update operations:

1. For file deletions: It is introduced an encrypted *deletion array* A_d, which will be used by the CS (based on the token provided by DU) for recovering the pointers to the nodes corresponding to the files that will be deleted. More specifically, for every file f, the A_d keeps a list L_f of nodes pointing to the nodes from A_s. Now, if f is deleted it is known that should be deleted from A_s. The authors call these corresponding nodes as *dual* nodes, denoted with N and $N*$.
2. For pointer modification: here the change is that the pointers within a node are encrypted with a homomorphic encryption scheme, letting the CS modify the encrypted pointers without working with decrypted values. Here, the authors used techniques based on XORing the message with an output of a pseudo-random function.
3. For memory management: in order to identify the free locations on A_s, a free list is used each time the server adds a node.

With the construction in [34], the authors have improved the initial scheme by adding dynamic operations and making the scheme CKS2-secure by using homomorphic encryption for pointer modification.

The process of transforming an SSE scheme into a DSE scheme involves adding mechanisms to handle updates to the encrypted data set. There are several approaches in this purpose used in research. Below are presented several techniques used in DSE construction from an SSE scheme:

- **Simple data structures**: One approach is to utilize tree-based data structures, such as B-trees or Merkle trees, to enable efficient updates and modifications to the encrypted dataset. This allows for dynamic insertion, deletion, and modification of records while preserving search functionality. Linked-list structures can be also used to handle updates in a DSE scheme. By maintaining a linked-list of encrypted records and associated pointers, efficient updates can be achieved. Another structure is the dynamic hash table, which by employing hash functions that support dynamic resizing and rehashing, the encrypted dataset can be efficiently updated.
- **Log-Structured Merge (LSM) Trees**: LSM trees are a type of data structure that is designed to handle large amounts of write operations. In the context of DSE, an LSM tree can be used to store the encrypted data set. When an update is made, it is first written to a buffer in memory. When the buffer is full, it is written to disk as a sorted array. These arrays are then periodically merged to maintain the search efficiency. This approach allows for efficient updates but requires additional mechanisms to ensure security.
- **Forward and Backward Privacy**: Forward privacy ensures that newly added documents are not searchable using old search tokens, while backward privacy ensures that deleted documents are not searchable using new search tokens. These properties can be achieved by using a unique identifier for each document and updating the search tokens whenever a document is added or deleted.

- **Partition-Based Methods**: In these methods, the encrypted data set is partitioned into several smaller subsets, each of which is managed independently. When an update is made, only the affected partition needs to be updated. This approach can significantly reduce the cost of updates, especially for large data sets.
- **Cryptographic Accumulators**: Cryptographic accumulators are a cryptographic primitive that can be used to maintain a set of elements and prove membership in the set. In the context of DSSE, cryptographic accumulators can be used to handle deletions in the encrypted data set. When a document is deleted, it is removed from the accumulator, and the search tokens are updated accordingly.
- **Two-Channel DSE**: In two-channel DSE, two separate communication channels are used: one for search queries and one for updates. This separation allows for more efficient handling of updates and can improve the overall performance of the DSE scheme.

DSS shows a viable path for safe and efficient data retrieval from encrypted databases. In fact, there is a library called OpenSSE[1] written in C++ that implements DSEs like Sophos [5], Diana [50], Janus [50], Thethis [56], and Pluto [56]. These schemes' dynamic nature allows for the insertion, deletion, and change of data after encryption, providing a high degree of flexibility and adaptability. DSSE systems have made tremendous progress in assuring data privacy and search efficiency, but difficulties remain. These include finding the best balance of search performance, update efficiency, and security assurances. Furthermore, implementing solid forward and backward security in DSSE systems is an important research topic. As we continue to investigate and improve DSSE, it is expected that these schemes will play an increasingly important role in safe data management, especially in cloud-based contexts where data privacy and rapid access are vital.

4.5.2 Homomorphic Searchable Encryption

The intersection of SE and HE has been a good resource for research, with the potential of unlocking new capabilities in secure data storage and computation. The combination of these two cryptographic primitives allows for the execution of complex queries and computations on encrypted data, therefore data privacy and security are ensured, while still enabling meaningful use of the data.

In the context of cloud computing, big data, and the Internet of Things, where large amounts of sensitive data are often stored and processed, the combination of SE and HE can provide enhanced security and functionality. For example, it allows a client to store encrypted data on an untrusted server and later retrieve specific portions of the data or perform computations on it without revealing any information about the data or the operations to the server.

However, combining SE and HE brings many challenges. It requires careful design to ensure that the combined system preserves the desirable properties of both SE and

[1] https://opensse.github.io.

HE. Moreover, the computational and communication overheads of the combined system must be manageable to make it practical for real-world applications.

There are several research works that have explored this combination. For example, the work [57] presents a homomorphic public key encryption scheme that allows the evaluation of 2-DNF formulas given an encrypted input. This work demonstrates the potential of combining SE and HE to perform complex computations on encrypted data. In another work [58], the authors propose a system that combines SE and HE to enable public auditability for data storage security in cloud computing. This system allows an external audit party to check the integrity of the outsourced data without demanding a copy of the entire data or introducing new vulnerabilities towards user privacy. Other work is [59]. This paper presents *Mash*, a fast genome and metagenome distance estimation method that extends the MinHash dimensionality-reduction technique to include a pairwise mutation distance and P-value significance test, enabling efficient clustering search of massive sequence collections.

An SE scheme that uses homomorphic encryption in its algorithms that gained the attention of the research community is [60], which developed an index-based SE method suitable for multi-user scenarios. This technique provides conjunctive keyword search and hides the search pattern using a specific Probabilistic HE (PHE) scheme. To efficiently implement the conjunctive keyword search functionality, the researchers used an auxiliary server and the additive homomorphic encryption technique, BCP [61]. This method guarantees that the DU only sees the desired search result. The auxiliary server is critical in allowing the system to accomplish the necessary features by leveraging what the authors refer to as the double trapdoor decryption process, which is a feature inherent in this architecture. The goal of [60] is to protect the search pattern while providing conjunctive keyword search capabilities. The idea behind this work is to utilize a polynomial representation of a multiset, by computing the polynomial representation $P_{w_i}(x)$ from $DB(w_i)$ for each keyword w_i. The polynomials are encrypted using the BCP cryptosystem to ensure their security. As a result, the search index I is formed by all the encrypted polynomials $Enc(P_{w_i}(x))$. When creating the trapdoor, each keyword in W is represented by an encrypted scalar. If the query Q contains the keyword w_i, the scalar r_i is a randomly generated integer otherwise it is 0, and the trapdoor for this term is $Ei = Enc(r_i)$. After getting the trapdoor, server C_1 conducts the search with the cooperation of server C_2. C_1 first computes $Enc(qi) = E_i \cdot_h Enc(P_{w_i}(x))$ with the assistance of C_2 using the multiplicative homomorphic property. Then, using the additive homomorphic condition, C_1 chooses a random polynomial $v_i(x)$ of appropriate degree and defines $Enc(Q_i(x)) = v_i(x) *_h Enc(q_i(x))$. Finally, based on the additive homomorphic condition, server C_1 provides to the user the encrypted polynomial $Enc(P(x)) = Enc(Q_1(x)) +_h \ldots +_h Enc(Q_M(x))$. The user reconstructs the polynomial $P(x)$ by decrypting the encrypted coefficients, computes its roots, and therefore retrieves the appropriate document IDs.

Further, we present the SH scheme proposed in [60] following the algorithm and notations used in the same paper:

- `Setup`$(1^K, DB)$: based on the secret parameter K and the entire database DB, the algorithm outputs the parameters of the system PP, the secret keys SK_{C_2} (for the second server C_2), SK_O (for the data owner), and the index structure I. Based on the algorithms of the BCP, BCP.`Setup` and BCP.`KeyGeneration`, the tuple (pp, msk, pk, sk) is obtained. The parameters of the system are defined as $PP = (pp, pk)$, the secret key $SK_{C_2} = msk$ and $SK_O = sk$.
  In order to generate the index structure I based on the indexes from DB, for every index w_i, is generated a polynomial $DB(w_i)$. Then, to the value $DB(w_i)$ there are appended a set of 0 values in order to achieve a predefined length L.
- `Trapdor`(W, Q, PP): in order to generate a trapdoor value based on the keywords from the query set $Q = \{w'_1, \ldots, w'_q\}$, DU needs to communicate with DO. So, DO receives the set Q from DU, and then generates the trapdoor $Tr = \{E_1, \ldots, E_m\}$. In this set, E_i is an encrypted scalar obtained as follows: DU checks whether $w_i \in W$ is in Q. If true, then DU generates randomly a value $r \in \mathbb{Z}_N$, else $r = 0$. Further, DO obtains the encrypted scalar $E_i \leftarrow$ `BCP.Encryption`(pk, r). Lastly, DO sends the trapdoor value to the first server C_1.
- `Search`(Tr, I, PP, SC_{K_2}): mainly the search algorithm is perfomed by the first server C_1, but it employs the second server C_2, which computes $Enc(q_i(x)) = E_i \cdot Enc(P_{w_i}(x))$, then generates randomly $v_i(x) \in \mathbb{R}^L[x]$, and achieves $Enc(Q_i(x)) = v_i(x) \cdot Enc(q_i(x))$. Lastly,, $Enc(P(x)) = Enc(Q_1(x) + \ldots + Q_M(x))$, polynomial that is sent to DU.
- `Decryption`$(SK_O, Enc(P(x)))$: DU decrypts the result helped by the DO, as follows: after the encrypted coefficients σ_i of the polynomial are decrypted, DU recovers the polynomial $P(x) = \sigma_{2L}x^{2L} + \cdots + \sigma_0$. The indexes that match the query keywords are the roots of the polynomial $P(x)$.

As we have seen, the authors use the polynomial representation of a multiset to provide the property of conjunctive keyword search. They protect the privacy of these polynomials by encrypting them using the BCP cryptosystem. The cloud server and an auxiliary server work together to do the search, taking use of both additive and multiplicative homomorphic features. However, the authors do not specify the sort of encryption strategy utilized to safeguard the papers before uploading them to the database.

Other works that include homomorphic properties in the SE schemes are [62–66].

A recent study on searchable encryption incorporating homomorphic properties is [67]. This manuscript offers an in-depth exploration of the progress in HE-based privacy-preserving methods, with a specific emphasis on their implementation in SE. The primary contributions of this study are the finding and categorization of existing SE frameworks that employ HE, a thorough examination of the types of HE utilized in SE, and an investigation into how HE influences the structure of the search process and facilitates additional features. The insights gathered indicate a rising trend in the application of HE in SE frameworks, particularly PHE. The widespread adoption of this category of HE frameworks, Paillier's cryptosystem in particular, can be ascribed to its straightforwardness, established security attributes,

Table 4.1 Preferred HE technique [67]

HE type	% of papers that use HE
PHE	57
SWHE	4
FHE	39

Table 4.2 Number of papers that uses HE techniques to achieve SE properties [67]

SE property	% of papers that used HE to achieve SE property
Regular expression/wildcard search	8.6
Conjunctive search	17.3
Disjunctive search	8.6
Range search	4.3
Phrase search	8.6
Ranked search	43.4
Verifiability	4.3
DSE	26.0

and extensive presence in open-source libraries. The analysis also underscores the dominance of index-based SE frameworks that use HE, the provision for ranked search and multi-keyword inquiries, and the necessity for further investigation into features such as verifiability and the capacity to authorize and revoke users. After applying the research methodology, in the paper are analyzed 23 papers that explicitly show the homomorphic properties of their proposed searchable encryption scheme (Table 4.1). The findings show that mostly, homomorphic encryption is used to achieve ranked search or to achieve a DSE. A summary of the findings from [67] can be seen in Table 4.2, while Table 4.1 shows which category of HE is mostly used. Note that some of the analyzed SE schemes may achieve multiple SE properties based on HE.

4.6 Case Studies and Applications

Searchable encryption has evolved as a powerful cryptographic technique with a wide range of applications in privacy-preserving data management and secure information retrieval. In this section, we will look at case studies and applications in different domains that highlight the adaptability and practicality of searchable encryption. By looking at these examples, we can see how searchable encryption methods may be used to protect sensitive data while allowing for quick and secure search operations. These case studies demonstrate the potential of searchable encryption to handle difficult data security and privacy issues across varied sectors, ranging from healthcare

to cloud storage, financial services to law enforcement. This section highlights the significance and continuous improvements of searchable encryption in addressing real-world challenges and supporting secure information transmission in nowadays digital ecosystems.

4.6.1 Cloud Computing and Big Data

Searchable encryption has a high potential in the domains of cloud computing and big data. Searchable encryption is a valuable tool as companies and organizations increasingly shift their data to the cloud and try to combat the security issues of maintaining and processing large amounts of data. It enables safe data storage on cloud servers while retaining the ability to execute relevant searches on the encrypted data. This is especially important in big data applications, where the ability to query massive databases rapidly and effectively is critical. Furthermore, searchable encryption guarantees that sensitive data stays private even if a security compromise occurs, adding an additional degree of security. As a result, the use of searchable encryption in cloud computing and big data not only improves data security but also allows effective data usage, making it a critical component in these fast-growing areas of the future. Figure 4.5 shows a summary of cloud computing and big data industries and domains of activities in which searchable encryption can be used in order to query data.

The objective of a recent research study [68] is to develop an advanced secure designated server public-key searchable encryption scheme that incorporates Multi-Ciphertext Integrity (MCI) and Multi-Trapdoor Privacy (MTP) based on bilinear pairings and Diffie-Hellman problem. The authors indicate that their system demonstrates superior efficiency in keyword ciphertext and trapdoor algorithms compared to the research literature mentioned in the paper. Nonetheless, their system is not resistant to multi-trapdoor attacks initiated by the server. This is due to the server's ability to formulate a specific equation using its private key, thereby discerning the relationship between multiple trapdoors.

Another study of SE with application in big data and cloud is [69]. This paper has several contributions: the proposed scheme has a high level of security, employing the user's private key for authentication, effectively resisting to inside keyword guessing attack (IKGA). Furthermore, the proposed scheme can achieve Multi-Trapdoor Privacy (MTP). On the other hand, the searchable encryption scheme achieves heterogeneity, by permitting both communication parties to operate in different environments, namely, Certificateless Cryptography (CLC) and Public Key Infrastructure (PKI), and use distinct cryptographic parameters. This enhances the practicality of the scheme. This study also uses bilinear pairing and the Computational Diffie-Hellman problem (CDHP).

An interesting approach is presented in [70], where authors used a clustering algorithm to group the documents. The document set is clustered using a clustering algorithm, resulting in multiple document clusters. On each cluster, a secure index

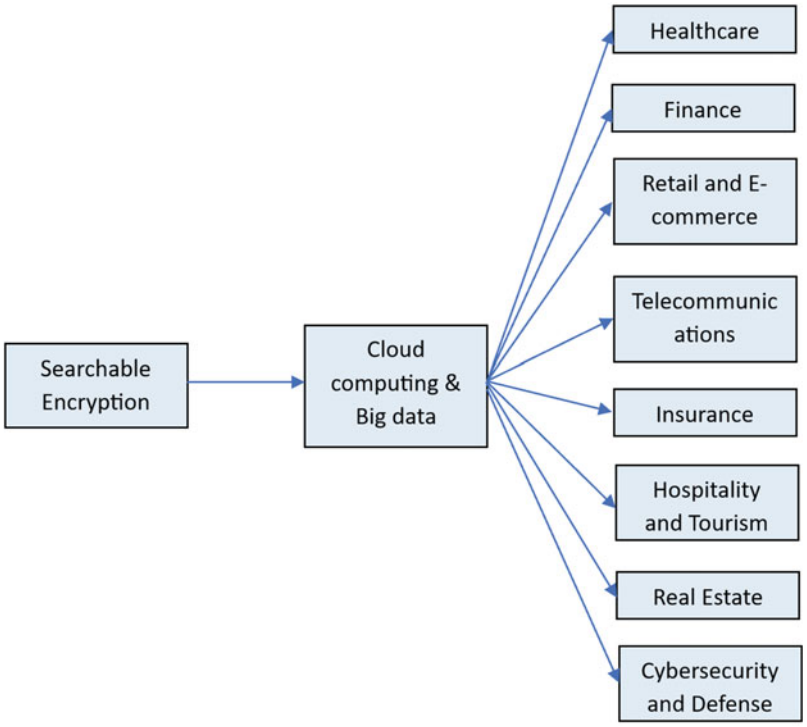

Fig. 4.5 Cloud computing and big data industries and domain of activities in which searchable encryption can be applied

tree is constructed. This approach reduces the time complexity of index tree retrieval by decreasing the height of the index tree. The search method in the proposed scheme is optimized such that it minimizes the number of index trees that need to be retrieved. This optimization significantly improves the query efficiency without compromising query accuracy too much. To encrypt the index and the query, the authors use an asymmetric scalar-product-preserving encryption, from which results a fast searchable symmetric encryption scheme that supports ranked search. Additionally, the scheme supports a dynamic update method that ensures the index in the scheme can safely accommodate document update operations. However, although the scheme supports sorting, the results cannot be verified.

In [71], the authors use machine learning algorithms in the search process. The authors present two effective strategies for ranked multi-keyword search over encrypted data, utilizing the k-means clustering technique. Their basic or enhanced scheme can yield the top-k most relevant results from all clusters, as opposed to a selected cluster, without compromising search accuracy. Additionally, the file clustering process in the enhanced scheme results in reduced file update overhead. Furthermore, the authors introduce a permutation matrix-based method for the dynamic

update of outsourced files. Their enhanced scheme is designed to achieve forward security, which safeguards against adversaries or cloud servers using previous query tokens to search newly added files.

We have seen that users involved in a system that uses SE are very active in working with data, a fact that makes SE being often used in big data and cloud computing together with authentication and authorization processes because it addresses the security and privacy concerns associated with these technologies, as follows:

1. Data Confidentiality: Searchable encryption allows authorized users to search and retrieve encrypted data without having to decrypt it. This ensures that sensitive information remains confidential, even during data processing and storage in the cloud.
2. Privacy Preservation: With searchable encryption, users can maintain the privacy of their data while still allowing authorized parties to perform searches and computations on the encrypted data. This is crucial in scenarios where the data owner wants to keep their data private but still needs to allow data analysis and processing.
3. Efficient Data Retrieval: Searchable encryption enables efficient retrieval of information from encrypted data. It allows for keyword-based searches and computations on encrypted data without revealing the actual content. This is particularly important in big data scenarios where large volumes of data need to be processed and analyzed quickly.
4. Access Control: By integrating authentication and authorization mechanisms with searchable encryption, only authorized users with proper credentials can access and perform searches on the encrypted data. This ensures that data is accessed and processed only by those who are authorized.

In SE, the DOs and DUs work intensely with the encrypted data. To gain more control over access, often, based on their identities or attributes, users may access specific data or trigger specific algorithms. Due to these aspects, often SE schemes are integrated or based on *Identity-based Encryption* (IBE) or *Attribute-based Encryption* (ABE) schemes. Further, we provide some examples of attribute-based searchable encryption schemes. Note that an IBE scheme, with the right procedures, often can be transformed into an ABE scheme according to [72].

Attribute-based Searchable Encryption Schemes

In [73] the authors propose an attribute-based searchable encryption scheme based on bilinear maps, called VMKS-ABE. The proposed scheme supports multi-keyword search while ensuring search privacy. This means that the CS can perform a search query based on multiple keywords using just one search trapdoor, but it cannot access any information about the searched keywords. Also, the search process is based on the DU's attributes. The computational cost of the proposed scheme is significantly reduced for the user client by outsourcing most of the computational tasks to the cloud proxy server. These tasks include private key generation, encryption, and decryption algorithms. Moreover, the outsourced private keys are verified, since the service

provider for outsourced private key generation is not entirely trusted. Therefore, the attribute authority needs to verify the correctness of the outsourced private keys to ensure they have been returned honestly. The security of the scheme is proven under the general group model. The authors prove that the keyword index is indistinguishable under adaptive keyword attacks, and the ciphertext is selectively secure against chosen plaintext attacks in the random oracle model.

Another research study that implements an attribute-based searchable encryption scheme is presented in [74]. Here, a novel approach based on Linear Secret Sharing Scheme (LSSS) matrix is introduced. This matrix enables the implementation of a multi-search mechanism and reduces the computational and storage costs by optimizing the calculation formula. To achieve a decrease in computational costs in the "range search" mechanism, this paper proposes the utilization of 0, 1-coding theory. This theory effectively reduces the number of numeric keywords in the searchable encryption area. The algorithms work as follows:

- `Setup`: Based on the security parameter λ, the pair of the public and private key is generated (PK, MSK), for which the private key is stored on the authorization center (CA).
- `KeyGeneration`: The private key SK for each data user is generated based on the MSK and a set of corresponding attributes.
- `Encrypt`: This algorithm encrypts a file f based on PK, an access policy Δ, and the set of keywords W associated with the file. The output is the encrypted file $CT_{\Delta,W}$, which is then stored CS.
- `Trapdoor`: Computes the trapdoor value based on the DU's SK and the searchable LSSS matrix (M, ρ).
- `Search`: The search algorithm is made based on the encrypted files and the trapdoor value. Firstly, the CS verifies whether the DU has the proper attributes in order to request the document. In the affirmative case, the CS sends to DU an encrypted result called mid-result (C, E).
- `Decrypt` Lastly, the DU decrypts the encrypted mid-result.

Another important research study is [75], in which the authors propose an approach to data querying using Ciphertext Policy Attributes Based Encryption (CP-ABE). The proposed scheme addresses three security requirements: confidentiality, queries over encrypted data, and flexible access control. By combining flexible access control and data confidentiality, CP-ABE can authenticate who can access data and possess the secret key. Further, in [76], the author presents a generic configurable stateless protocol for secure attribute-based authentication, authorization, storage, and transmission in distributed storage systems based upon CP-ABE.

4.6.2 Internet of Things

In the IoT ecosystem, devices continuously generate and send data to data centers. The data, if not properly secured, can be exploited, leading to privacy breaches and

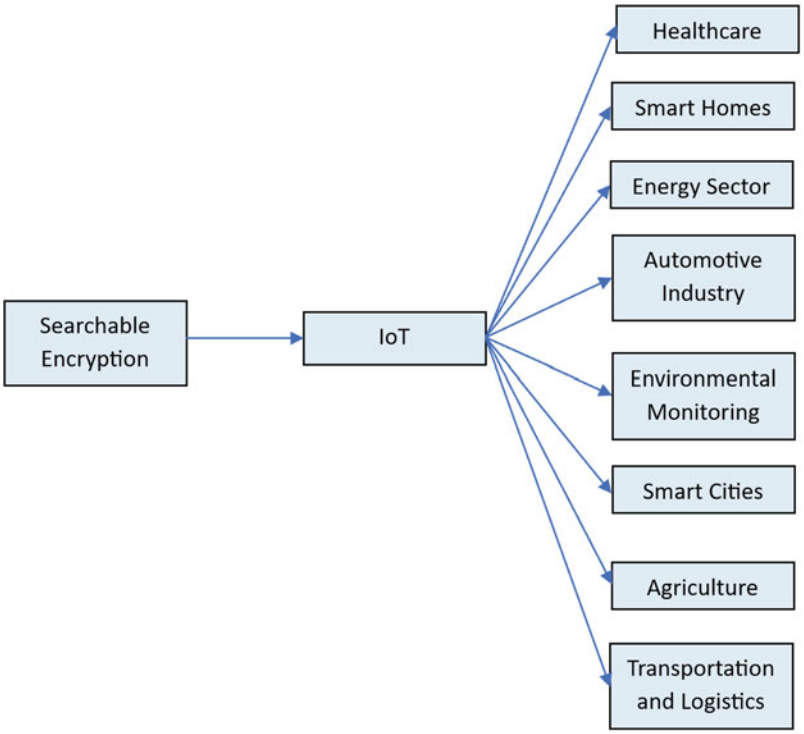

Fig. 4.6 IoT industries and domain of activities in which searchable encryption can be applied

data leakage. SE can provide a powerful solution to this problem by allowing data to be stored and transmitted in an encrypted form, while still permitting efficient search operations. This ensures that the data remains secure and private, even in the event of unauthorized access.

Usually, IoT devices have limited computational resources and power. Traditional encryption methods, which require decryption before data can be searched, can be resource-consuming and impractical for IoT devices. SE, on the other hand, allows for efficient search operations on encrypted data, making it an ideal solution for resource-constrained IoT devices.

SE also enables secure data sharing in the IoT. Multiple users can search the encrypted data without compromising its security, which is particularly important in multi-user IoT environments. For instance, in a smart home scenario, different family members can search the data generated by smart devices without revealing the data to each other or to potential eavesdroppers. Figure 4.6 shows a summary of IoT industries and domains of activities in which searchable encryption can be used in order to query data.

In [77], the authors have developed an online/offline Ciphertext-Policy Attribute-Based Keyword Search (CP-ABKS) scheme for the Industrial Internet of Things

(IIoT). This scheme reduces the computational cost of the algorithm through the use of a reusable ciphertext pool and outsourcing decryption. The scheme supports efficient ciphertext verification and can dynamically control user access through user attributes. It has been proven to provide IND-sAA-sCCA2 security. Given the openness of the IIoT, the authors' scheme can protect sensitive information when integrated with the IIoT environment. The entities involved in the system are: Central Authority (CA), Attribute Authority (AA), Industrial Cloud (INC), Edge Service Provider (ESP), Data Owner (DO), Data User (DU). The proposed system works as follows: CA initially runs the `GlobalSetup` algorithm to generate public parameters PP and master key MK. Then, AA computes the key pairs (SK_{AA}, PK_{AA}) and INC will use GP to obtain (SK_s, PK_s). Finally, the AA adds the secret key $SK_{uid,a}$ and attribute key AK_a for a user and attribute a, marking the end of the scheme's initialization phase. In the encryption phase, the authors first execute a `CPoolGen` algorithm to obtain $CPool$ for reuse. The DO runs `OffEnc` and `OnEnc` to compute CT, but the `OffEnc` algorithm is executed only once for a message m. The DO then defines a keyword set W and generates IDX. There are allowed all roles in the system to verify the ciphertext to ensure the scheme's security. After the DO completes the encryption, the ciphertext is sent to the edge server. The can then use the attribute key AKa to re-encrypt. When the DU needs to decrypt a message, they first perform the `Trapdoor` algorithm to generate TP and send it to INC. The INC then runs the `Search` algorithm to match the ciphertext. When the cloud sends the ciphertext to ESP, the DU obtains a blind key SK and sends it to ESP for pre-decryption. In the end, the DU performs `FullDec` to retrieve the message.

4.6.3 Blockchain

Searchable encryption, when integrated with blockchain technology, opens up lots of possibilities for secure data management and retrieval. Blockchain's inherent characteristics of decentralization, immutability, and transparency can significantly enhance the security and reliability of searchable encryption schemes. For example, blockchain can provide a tamper-proof environment for storing encrypted indexes or keys, thereby ensuring the integrity and authenticity of search operations. Furthermore, the decentralized nature of blockchain can overcome the risks associated with single-point failures or centralized attacks, enhancing the robustness of the searchable encryption system. The integration of these two technologies can be particularly beneficial in applications such as secure data sharing, privacy-preserving data marketplaces, and confidential data management in distributed networks. However, the practical implementation of such systems requires careful consideration of factors like computational efficiency, scalability, and user privacy. Figure 4.7 shows a summary of Blockchain industries and domains of activities in which searchable encryption can be used in order to query data.

In [78] the authors present three contributions, as follows: (i) Initially, they develop a public-key authenticated encryption scheme that is based on blockchain

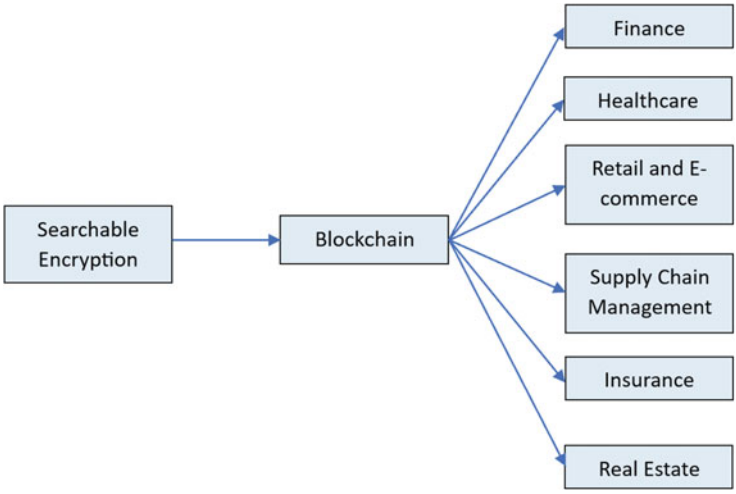

Fig. 4.7 Blockchain industries and domain of activities in which searchable encryption can be applied

technology. The data sender and receiver can naturally share a common initial key via the Diffie-Hellman key agreement, which is resistant to inside keyword guessing attacks (IKGA). This scheme ensures that the search and access patterns are not compromised. (ii) Then, the authors prove that their scheme significantly reduces computational costs. The scheme's practicality is based on the fact that it minimizes the computational cost of ciphertext and trapdoor generation without the need for bilinear pairs and during the testing phase, the inner product relationship between two vectors is used in order to determine set inclusion. (iii) Lastly, the proposed scheme supports the verification of result integrity. By including blockchain, the reliability of search results is ensured. Moreover, the authors demonstrate that the scheme provides adaptive security for CKA, and conduct a series of experiments to evaluate its performance. In this work, the hardness assumption based on which the scheme is constructed is Decisional Oracle Diffie-Hellman (DOFH), and blockchain is used to store data and verify the trapdoor value sent by data receiver. In short, the process is as follows [78]:

1. Data sender extracts the keyword set W from the documents Doc and constructs a checklist B from the files. Simultaneously, it encrypts the keyword index C_w and the documents C_{Doc}.
2. Data sender sends the encrypted keyword index C_w to the Service Provider for storage. At the same time, the data sender sends the checklist Acc to the Blockchain.
3. Service Provider registers C_w and C_{Doc} in the database.
4. Data receiver computes the trapdoor T_w of the queried keywords W_q.
5. Data receiver sends the trapdoor T_w to the service provider and the Blockchain.

6. Service Provider searches for the keyword index C_w in the database using the trapdoor T_w.
7. Service Provider returns the result to the DR and the Blockchain for verification.
8. As soon as the Service Provider returns the results, the Blockchain calculates the hash value of each result. From the identity information of the data sender in the checklist Acc, the Blockchain retrieves the corresponding hash values of the files. Finally, the Blockchain compares the two hash values to generate the proof.
9. The Blockchain sends the proof to the DR.

Note in the description from above that the data sender is equivalent to the DO, the data receiver is equivalent to the DU, and the service provider is equivalent to CS, components presented in Sect. 4.2.1, and additionally, the scheme [78] involves a blockchain network. However, the authors expand their construction to a more efficient form and they prove the scheme is IND-CKA and IND-KGA secure.

Another recent work that uses blockchain in a SE environment is [79]. Here, the authors bring the following contributions: (i) presentation of a blockchain-based multiuser normalized searchable encryption (BNSE) system that provides fast ciphertext data retrieval in multiuser settings and facilitates ciphertext data supervision. (ii) Based on the methodology mentioned in (i), they create a blockchain-based normalized searchable encryption system for medical data (BNSEM), which enables the use of retrieval on encrypted medical data. The scheme works is based on the Decisional Bilinear Diffie-Hellman (DBDH) problem and it works as follows:

1. The DO initially encrypts the data file using a symmetric encryption algorithm, followed by encrypting the symmetric key using a public key cryptography algorithm. Finally, the DO uploads the ciphertext to the CS.
2. The DO extracts keyword-index pairs from the files and generates searchable encrypted data structures. These structures are then uploaded to the blockchain.
3. The DU generates a search trapdoor containing the target keyword and sends the search request, which includes the trapdoor, to a nearby blockchain node. This search request triggers the search process within the smart contract, which subsequently returns the index of all matching encrypted files.
4. The DU first decrypts the encrypted file index returned by the smart contract in the previous step. After obtaining the plaintext index, the DU accesses the data in the CS.
5. Based on the file indexes submitted by the DU, the CS returns the corresponding files.

The authors prove their proposed scheme is forward and backward private and then integrate this process in an electronic medical system and prove its applicability.

Another approach for searchable encryption blockchain-based is presented in [80], where authors include control access based on the user' attributes. The authors have implemented an access control method based on attributes, utilizing smart contracts. The DO establishes an access policy matrix through Linear Secret-Sharing Schemes (LSSS), and calculates the inner product with a column vector constructed by a secret value s. The resulting access control vector is then transferred to the

blockchain. To enable search functionality, the DUs first invoke the Access Control Contract (ACC) with their own attributes. Only DUs who meet the access policy can obtain a valid verification code, which they can then use to search and retrieve the corresponding documents. Further, the authors also implemented an enhanced searchable encryption method based on smart contracts. The proposed scheme allows the DO to go offline freely after the process of creating the index structure. Moreover, the DU can customize the search token for searching, requiring few resources. After obtaining the document label, the DU downloads the corresponding ciphertext from the cloud database, while simultaneously submitting a transaction to a trusted Key Management Server (KMS) for verification. Once the KMS verifies the transaction, it returns the symmetric key of the ciphertext document through a secure channel, enabling the DU to decrypt the document. With attribute-based access control, DUs that comply with the access policy can ask of the search service without the need to pay deposits for each search query. The assurance of fairness is derived from the audibility and authenticity of the process of attributes verification implemented on the ACC. The authors prove that the proposed scheme is secure against $\mathcal{L}$ - nonadaptive attacks, where $\mathcal{L}$ is the leakage function.

Due to its transparent, secure, tamper-proof, and flexible ledger services, blockchain technology is rapidly integrated in a variety of industries, including cloud computing, banking, supply chain, and healthcare [81]. Several research studies have investigated the integration of blockchain with searchable encryption, addressing issues such as operational expense, privacy loss, and security concerns [82]. Another important research [83] presented a feasible architecture for storing and searching encrypted data on cloud storage. The framework, which is called CryptoSearch, has proven a safe search process on encrypted data and provides a good performance in real-world applications. Another study focuses on the application of AI and blockchain to protect privacy [84]. The research focused on the use of blockchain in particular application situations such as data encryption , de-identification, multi-tier distributed ledgers, and k-anonymity approaches. A recent survey [85] shows that Ethereum is preferred in searchable encryption blockchain-based schemes, and smart contract is preferred in the verification process, followed by the verification made on the client side. The same study shows that mostly, the implementations fall in the SSE category, rather than PKSE.

4.7 Challenges and Research Directions

Searchable encryption is a study area that tries to securely outsource data to a third-party server while enabling users to search the encrypted data. With the examples and case studies presented in this section, we can see that searchable encryption can be used in real-life applications. However, it did not reach a maturity level such that to be used on a large scale, therefore, further research is needed. Below, we present some of the main research directions in searchable encryption:

1. Efficient and scalable search operations over encrypted data: One of the key issues in searchable encryption is achieving efficient and scalable search operations over encrypted data. The research focuses on creating approaches that reduce computing overhead while improving search speed, particularly for large-scale databases.
2. Privacy and security: A major research direction is to protect the privacy and security of both the data owner and the person doing the search. To improve the privacy and security guarantees of searchable encryption systems, techniques such as trapdoor indistinguishability, oblivious transfer, and safe multi-party computing are investigated.
3. Query expressivity: Traditional searchable encryption methods often allow restricted search capability, such as precise keyword matching or conjunctive queries. More expressive search features, such as range queries, ranked retrieval, and complicated searches involving several keywords and logical operators, are being researched.
4. Dynamic data and updates: Another continuing research topic is enabling quick changes to encrypted data while retaining searchability. Dynamic symmetric searchable encryption (DSSE) and dynamic public key searchable encryption (DPKSE) are techniques that attempt to facilitate quick insertion, deletion, and modification operations on encrypted data.
5. Leakage and Side-Channel Attacks: Searchable encryption is vulnerable to leakage attacks, in which an adversary uses information leakage from search operations to deduce sensitive information. The research focuses on detecting and mitigating such side-channel assaults by creating leakage-resistant encryption schemes and testing their resilience against diverse attacks.
6. Practical deploy techniques: A critical research area is translating theoretical advances into practical and deployable solutions. Real-world restrictions like as performance, usability, compatibility with current systems, and interaction with popular cloud platforms must all be considered.
7. Hybrid Approaches: To achieve a balance of security, privacy, and efficiency in searchable encryption systems, researchers are investigating hybrid approaches that combine different cryptographic techniques such as secure multi-party computation, homomorphic encryption, and searchable symmetric encryption.
8. Verifiability and Provable Security: Formal security models, provable security guarantees, and verifiability techniques are key research fields. Security proofs serve in establishing the reliability of searchable encryption methods and in the construction of standardized protocols [48, 49].

References

1. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000, pp. 44–55. IEEE (2000)
2. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2–6, 2004. Proceedings, vol. 23, pp. 506–522. Springer (2004)
3. Boldyreva, A., Chenette, N., Lee, Y., O'neill, A.: Order-preserving symmetric encryption. In: Advances in Cryptology-EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26–30, 2009. Proceedings, vol. 28, pp. 224–241. Springer (2009)
4. Kamara, S., Lauter, K.: Cryptographic cloud storage. In: International Conference on Financial Cryptography and Data Security, pp. 136–149. Springer (2010)
5. Bost, R.: $\Sigma o\phi o\xi$: forward secure searchable encryption. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1143–1154 (2016)
6. Pham, H., Woodworth, J., Amini Salehi, M.: Survey on secure search over encrypted data on the cloud. Concurr. Comput. Pract. Exp. **31**, e5284 (2019)
7. Wang, Y., Wang, J., Chen, X.: Secure searchable encryption: a survey. J. Commun. Inf. Netw. **1**, 52–65 (2016)
8. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Trans. Parallel Distrib. Syst. **25**, 222–233 (2013)
9. Fu, Z., Sun, X., Liu, Q., Zhou, L., Shu, J.: Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing. IEICE Trans. Commun. **98**, 190–200 (2015)
10. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 79–88 (2006)
11. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: International Conference on Applied Cryptography and Network Security, pp. 442–455. Springer (2005)
12. Xia, Z., Zhu, Y., Sun, X., Chen, L.: Secure semantic expansion based search over encrypted cloud data supporting similarity ranking. J. Cloud Comput. **3**, 1–11 (2014)
13. Wang, C., Cao, N., Ren, K., Lou, W.: Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Trans. Parallel Distrib. Syst. **23**, 1467–1479 (2011)
14. Moh, T.-S., Ho, K.H.: Efficient semantic search over encrypted data in cloud computing. In: 2014 International Conference on High Performance Computing & Simulation (HPCS), pp. 382–390. IEEE (2014)
15. Saleem, M., Warsi, M.R., Khan, N.S.: Secure metadata based search over encrypted cloud data supporting similarity ranking. Int. J. Comput. Sci. Inf. Secur. **15**, 353 (2017)
16. Baek, J., Safavi-Naini, R., Susilo, W.: Public Key encryption with keyword search revisited. In: Gervasi, O. et al. (eds.) Computational Science and Its Applications—ICCSA 2008, pp. 1249–1259. Springer (2008)
17. Xia, Z., Wang, X., Sun, X., Wang, Q.: A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Trans. Parallel Distrib. Syst. **27**, 340–352 (2015)
18. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Theory of Cryptography: 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21–24, 2007. Proceedings, vol. 4, pp. 535–554. Springer (2007)
19. Goh, E.-J.: Secure indexes. Cryptology ePrint Archive (2003)
20. Ding, M., Gao, F., Jin, Z., Zhang, H.: An efficient public key encryption with conjunctive keyword search scheme based on pairings. In: 2012 3rd IEEE International Conference on Network Infrastructure and Digital Content, pp. 526–530. IEEE (2012)

21. Yang, Y., Li, H., Liu, W., Yao, H., Wen, M.: Secure dynamic searchable symmetric encryption with constant document update cost. In: 2014 IEEE Global Communications Conference, pp. 775–780. IEEE (2014)
22. Byun, J.W., Rhee, H.S., Park, H.-A., Lee, D.H.: Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In: Workshop on Secure Data Management, pp. 75–83. Springer (2006)
23. Lu, Y., Wang, G., Li, J.: Keyword guessing attacks on a public key encryption with keyword search scheme without random oracle and its improvement. Inf. Sci. **479**, 270–276 (2019)
24. Fang, L., Susilo, W., Ge, C., Wang, J.: Public key encryption with keyword search secure against keyword guessing attacks without random oracle. Inf. Sci. **238**, 221–241 (2013)
25. Zhang, Y., Katz, J., Papamanthou, C.: All your queries are belong to us: the power of File-Injection attacks on searchable encryption. In: 25th USENIX Security Symposium (USENIX Security 16), pp. 707–720 (2016)
26. Di Crescenzo, G., Saraswat, V.: Public key encryption with searchable keywords based on Jacobi symbols. In: Progress in Cryptology–INDOCRYPT 2007: 8th International Conference on Cryptology in India, Chennai, India, December 9–13, 2007. Proceedings, vol. 8, pp. 282–296. Springer (2007)
27. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Cryptography and Coding: 8th IMA International Conference Cirencester, UK, December 17–19, 2001. Proceedings, vol. 8, pp. 360–363. Springer (2001)
28. Jacobi, C.G.J.: Über die Kreistheilung und ihre Anwendung auf die Zahlentheorie. (1846)
29. Khader, D.: Public key encryption with keyword search based on K-resilient IBE. In: International Conference on Computational Science and Its Applications, pp. 298–308. Springer (2006)
30. Van Liesdonk, P., Sedghi, S., Doumen, J., Hartel, P., Jonker, W.: Computationally efficient searchable symmetric encryption. In: Secure Data Management: 7th VLDB Workshop, SDM 2010, Singapore, September 17, 2010. Proceedings, vol. 7, pp. 87–100. Springer (2010)
31. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13**, 422–426 (1970)
32. Ren, S.Q., et al.: Secure searching on cloud storage enhanced by homomorphic indexing. Futur. Gen. Comput. Syst. **65**, 102–110 (2016)
33. Liu, C., Zhu, L., Chen, J.: Efficient searchable symmetric encryption for storing multiple source dynamic social data on cloud. J. Netw. Comput. Appl. **86**, 3–14 (2017)
34. Kamara, S., Papamanthou, C.: Parallel and dynamic searchable symmetric encryption. In: Financial Cryptography and Data Security: 17th International Conference, FC 2013, Okinawa, Japan, April 1–5, 2013, Revised Selected Papers, vol. 17, pp. 258–274. Springer (2013)
35. Kamara, S., Papamanthou, C.: Parallel and dynamic searchable symmetric encryption. In: Financial Cryptography and Data Security: 17th International Conference, FC 2013, Okinawa, Japan, April 1–5, 2013, Revised Selected Papers vol. 17, pp. 258–274. Springer (2013)
36. Salehi, M.A. et al.: Reseed: regular expression search over encrypted data in the cloud. In: 2014 IEEE 7th International Conference on Cloud Computing, pp. 673–680. IEEE (2014)
37. Salehi, M.A. et al.: RESeED: a secure regular-expression search tool for storage clouds. Softw. Pract. Exp. **47**, 1221–1241 (2017)
38. Fu, Z., Wu, X., Guan, C., Sun, X., Ren, K.: Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. IEEE Trans. Inf. Forens. Secur. **11**, 2706–2716 (2016)
39. Wang, B., Yu, S., Lou, W., Hou, Y.T.: Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. In: IEEE INFOCOM 2014—IEEE Conference on Computer Communications, pp. 2112–2120. IEEE (2014)
40. Li, J. et al.: Fuzzy keyword search over encrypted data in cloud computing. In: 2010 Proceedings IEEE INFOCOM, pp. 1–5. IEEE (2010)
41. Liu, C., Zhu, L., Li, L.: Fuzzy keyword search on encrypted cloud storage data with small index. In: 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, pp. 269–273. IEEE (2011)

42. Wang, J., et al.: Efficient verifiable fuzzy keyword search over encrypted data in cloud computing. Comput. Sci. Inf. Syst. **10**, 667–684 (2013)
43. Lovins, J.B.: Development of a stemming algorithm. Mech. Transl. Comput. Linguist. **11**, 22–31 (1968)
44. Durrani, N., Schmid, H., Fraser, A., Koehn, P., Schütze, H.: The operation sequence model—combining n-gram-based and phrase-based statistical machine translation. Comput. Linguist. **41**, 185–214 (2015)
45. Sun, X., Zhu, Y., Xia, Z., Chen, L.: Privacy-preserving keyword-based semantic search over encrypted cloud data. Int. J. Secur. Appl. **8**, 9–20 (2014)
46. Woodworth, J., Salehi, M.A., Raghavan, V.: S3C: an architecture for space-efficient semantic search over encrypted data in the cloud. In: 2016 IEEE International Conference on Big Data (Big Data), pp. 3722–3731. IEEE (2016)
47. Yang, Y., Zheng, X., Chang, V., Tang, C.: Semantic keyword searchable proxy re-encryption for postquantum secure cloud storage. Concurr. Comput. Pract. Exp. **29**, e4211 (2017)
48. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 965–976 (2012)
49. Stefanov, E., Papamanthou, C., Shi, E.: Practical dynamic searchable encryption with small leakage. Cryptology ePrint Archive (2013)
50. Bost, R., Minaud, B., Ohrimenko, O.: Forward and backward private searchable encryption from constrained cryptographic primitives. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1465–1482 (2017)
51. Sun, S.-F. et al.: Practical backward-secure searchable encryption from symmetric puncturable encryption. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 763–780 (2018)
52. Green, M.D., Miers, I.: Forward secure asynchronous messaging from puncturable encryption. In: 2015 IEEE Symposium on Security and Privacy, pp. 305–320. IEEE (2015)
53. Sun, S.-F. et al.: Practical non-interactive searchable encryption with forward and backward privacy. In: NDSS (2021)
54. Zhang, H., Zeng, S., Yang, J.: Backward private dynamic searchable encryption with update pattern. Inf. Sci. **624**, 1–19 (2023)
55. Peng, Y. et al.: Dynamic searchable symmetric encryption with forward and backward privacy. In: 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 420–427. IEEE (2021)
56. Bossuat, A., Bost, R., Fouque, P.-A., Minaud, B., Reichle, M.: SSE and SSD: page-efficient searchable symmetric encryption. In: Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021. Proceedings, Part III, vol. 41, pp. 157–184. Springer (2021)
57. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Theory of Cryptography: Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10–12, 2005. Proceedings, vol. 2, pp. 325–341. Springer (2005)
58. Wang, C., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for data storage security in cloud computing. In: 2010 Proceedings IEEE INFOCOM, pp. 1–9. IEEE (2010)
59. Ondov, B.D., et al.: Mash: fast genome and metagenome distance estimation using MinHash. Genome Biol. **17**, 1–14 (2016)
60. Wang, Y., Sun, S.-F., Wang, J., Liu, J.K., Chen, X.: Achieving searchable encryption scheme with search pattern hidden. IEEE Trans. Serv. Comput. **15**, 1012–1025 (2022)
61. Bresson, E., Catalano, D., Pointcheval, D.: A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 37–54. Springer (2003)
62. Liu, G., Yang, G., Bai, S., Wang, H., Xiang, Y.: FASE: a fast and accurate privacy-preserving multi-keyword top-k retrieval scheme over encrypted cloud data. IEEE Trans. Serv. Comput. **15**, 1855–1867 (2020)

63. Andola, N., Prakash, S., Yadav, V.K., Venkatesan, S., Verma, S.: A secure searchable encryption scheme for cloud using hash-based indexing. J. Comput. Syst. Sci. **126**, 119–137 (2022)
64. Prakash, A.J., Elizabeth, B.L.: Pindex: private multi-linked index for encrypted document retrieval. PLoS One **16**, e0256223 (2021)
65. Hou, J., Liu, Y., Hao, R.: Privacy-Preserving Phrase Search over Encrypted Data. In: Proceedings of the 4th International Conference on Big Data Technologies, pp. 154–159 (2021)
66. Li, Y., Zhou, F., Xu, Z., Ge, Y.: An efficient two-server ranked dynamic searchable encryption scheme. IEEE Access **8**, 86328–86344 (2020)
67. Amorim, I., Costa, I.: leveraging searchable encryption through homomorphic encryption: a comprehensive analysis. Mathematics **11**, 2948 (2023)
68. Tian, C., Yang, G., Guo, J., Han, L., Liu, X.: An improved secure designated server public key searchable encryption scheme with multi-ciphertext indistinguishability. J. Cloud Comput. **11** (2022)
69. Luo, M., Huang, D., Qiu, M.: An enhanced heterogeneous public key searchable encryption scheme supporting multiple keywords. Peer-to-Peer Netw. Appl. **16**, 383–394 (2023)
70. He, W., Li, Y., Zhang, Y.: Fast, Searchable, Symmetric Encryption Scheme Supporting Ranked Search. Symmetry, **14** (2022)
71. Miao, Y., et al.: Ranked keyword search over encrypted cloud data through machine learning method. IEEE Trans. Serv. Comput. **16**, 525–536 (2022)
72. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 89–98 (2006)
73. Wang, S., Jia, S., Zhang, Y.: Verifiable and multi-keyword searchable attribute-based encryption scheme for cloud storage. IEEE Access **7** (2019)
74. Li, Y., Wang, H., Wang, S., Ding, Y.: Attribute-Based Searchable Encryption Scheme Supporting Efficient Range Search in Cloud Computing. In: 2021 IEEE Conference on Dependable and Secure Computing (DSC), pp. 1–8 (2021)
75. Almarwani, M., Konev, B., Lisitsa, A.: Data querying with ciphertext policy attribute based encryption (2022). arXiv preprint arXiv:2209.15103
76. Alston, A.: Attribute-based encryption for attribute-based authentication, authorization, storage, and transmission in distributed storage systems (2017). arXiv preprint arXiv:1705.06002
77. Niu, S., Hu, Y., Su, Y., Yan, S., Zhou, S.: Attribute-based searchable encrypted scheme with edge computing for industrial Internet of Things. J. Syst. Arch. **139**, 102889 (2023)
78. Peng, C. et al.: A Lightweight blockchain-based public-key authenticated encryption with multi-keyword search for cloud computing. Secur. Commun. Netw. **2022** (2022)
79. Peng, C. et al.: A Blockchain-based normalized searchable encryption system for medical data. Secur. Commun. Netw. **2022** (2022)
80. Wang, H. et al.: Attribute-based access control meets blockchain-enabled searchable encryption: a flexible and privacy-preserving framework for multi-user search. Electronics **11** (2022)
81. Li, X., Wu, W.: Recent advances of blockchain and its applications. J. Soc. Comput. **3**, 363–394 (2022)
82. Li, Z. et al.: An Overview of AI and blockchain integration for privacy-preserving (2023). arXiv preprint arXiv:2305.03928
83. Islam, M.: A practical framework for storing and searching encrypted data on cloud storage (2023). arXiv preprint arXiv:2306.03547
84. Tang, Q.: Towards blockchain-enabled searchable encryption. In: Information and Communications Security: 21st International Conference, ICICS 2019, Beijing, China, December 15–17, 2019, Revised Selected Papers, vol. 21, pp. 482–500. Springer (2020)
85. How, H.-B., Heng, S.-H.: Blockchain-enabled searchable encryption in clouds: a review. J. Inf. Secur. Appl. **67**, 103183 (2022)

Index

S. L. Nita and M. I. Mihailescu, *Advances to Homomorphic and Searchable Encryption*, https://doi.org/10.1007/978-3-031-43214-9