Department of Computer Science
University of El Oued, Algeria

Efficient Data Analysis Based on Machine Learning Techniques, Application to IoT (Management and Security)

A dissertation submitted on December, 2024 for the degree of
Doctor of Science

By
Khaled Chait

Directed by
Prof. Abdelkader Laouid

Alhamdulillah

This work is dedicated wholeheartedly
to the memory of my father,
and to my entire family...

Acknowledgments

I want to express my sincere thanks to Professor Abdelkader Laouid, my thesis director and close friend. From the outset, he provided invaluable guidance, ensuring that I navigated the correct and unequivocally assured path towards achieving my goal. I am profoundly grateful for his generous sharing of knowledge, insightful advice, unwavering patience, consistent availability, and steadfast support throughout these years.

I extend my sincere appreciation to my mentors and colleagues, Prof. Mohammad Hammoudeh, Saudi Aramco Cybersecurity Chair Professor at King Fahd University of Petroleum and Minerals, KSA, and Dr. Mostefa Kara, Postdoctoral fellow at King Fahd University of Petroleum and Minerals, KSA. Their guidance and unwavering enthusiasm have been instrumental in shaping my journey, and I am deeply grateful for their valuable support.

I also want to share my deepest thanks to all the esteemed jury members who played a pivotal role in evaluating my thesis. Special thanks to Prof. Mohammad Charaf Eddine Meftah, Full Professor at the University of El Oued, Algeria, for the honor of chairing my thesis jury. Additionally, I am thankful to Dr. Ismail Kertiou, Associate Professor at the University of El Oued, Algeria, Dr. Afifa Ghenai, Associate Professor at the University of Constantine 2, Algeria, Dr. Lakhdar Laimeche, Associate Professor at the University of Tebessa, Algeria, and Dr. Ahmed Ahmim, Associate Professor at the University of Souk Ahras, Algeria. I am sincerely appreciative of their willingness to serve as judges for my thesis and for the keen interest they have demonstrated in evaluating my work. Their expertise and commitment are invaluable, and I am privileged to have benefited from their collective wisdom in the examination of my research.

I extend my heartfelt gratitude to all individuals whose encouragement significantly contributed to the successful completion of my doctoral thesis. Your support has been invaluable, and I deeply appreciate the positive impact each one of you has had on this significant achievement. Thank you for being a crucial part of this journey.

Finally, I want to thank my loved ones—my mother, wife, daughter, brother, sisters, and the entire family—for their unwavering support. This achievement wouldn't have been possible without them.

Khaled Chait

Abstract

Efficient Data Analysis Based on Machine Learning Techniques, Application to IoT (Management and Security)

In today's world, all enterprises generate massive amounts of data from diverse sources. Whether it's from enterprise systems themselves, from social media or other online sources, from smartphones and other client/edge computing devices, or from sensors and instruments comprising the Internet of Things (IoT), this data is extremely valuable to organizations that have the tools in place to capitalize on it. The overall toolbox for these tools is called data analytics. The latter can be challenging, given that it involves complex information and large data sets that are difficult to manage manually. This is where machine learning (ML) comes in. It is changing the way consumer-based companies are dealing with the vast data they generate. ML is based on Artificial Intelligence (AI) on the idea that systems can automatically learn from data, identify patterns, and make decisions with little human interference, and is being used to automate analytical model building.

Data has become a key driver for enterprises to enhance their sustainability in a competitive business world. With more data being produced and stored than ever before, the need for more efficient, effective, and precise processes has grown too. ML is seen to be a great solution, it significantly reduces efforts, saves time and is a cost-effective tool which replaces multiple teams working on analyzing, processing and performing regression testing on the data. It gives accurate results and helps organizations build statistical models based on real-time data.

In this thesis, the goal is to design and implement efficient machine learning-based solutions in order to help make better data analysis. The context of IoT will be mainly considered as management and security domains are bringing some of the biggest challenges in this area.

Keywords: Machine learning, Artificial intelligence, Data analysis, IoT, Management and security.

Résumé

Analyse Efficace des Données Basée sur des Techniques d'Apprentissage Automatique, Application à l'IoT (Gestion et Sécurité)

Dans le monde d'aujourd'hui, toutes les entreprises génèrent d'énormes quantités de données provenant de sources diverses. Qu'elles proviennent des systèmes d'entreprise eux-mêmes, des médias sociaux ou d'autres sources en ligne, des smartphones et autres appareils informatiques clients/edge computing, ou des capteurs et instruments composant l'Internet des objets (IoT), ces données sont extrêmement précieuses pour les organisations qui disposent des outils nécessaires. en place pour en tirer profit. La boîte à outils globale de ces outils s'appelle l'analyse des données. Cette dernière solution peut s'avérer difficile, car elle implique des informations complexes et de vastes ensembles de données difficiles à gérer manuellement. C'est là qu'intervient l'apprentissage automatique (ML). Il change la façon dont les entreprises axées sur le consommateur traitent les vastes données qu'elles génèrent. Le ML est basé sur l'intelligence artificielle (IA) sur l'idée que les systèmes peuvent automatiquement apprendre des données, identifier des modèles et prendre des décisions avec peu d'interférence humaine, et est utilisé pour automatiser la création de modèles analytiques.

Les données sont devenues un facteur clé permettant aux entreprises d'améliorer leur durabilité dans un monde des affaires compétitif. Avec plus de données produites et stockées que jamais auparavant, le besoin de processus plus efficients, efficaces et précis s'est également accru. Le ML est considéré comme une excellente solution, il réduit considérablement les efforts, fait gagner du temps et constitue un outil rentable qui remplace plusieurs équipes travaillant sur l'analyse, le traitement et la réalisation de tests de régression sur les données. Il donne des résultats précis et aide les organisations à créer des modèles statistiques basés sur des données en temps réel.

Dans cette thèse, l'objectif est de concevoir et de mettre en œuvre des solutions efficaces basées sur l'apprentissage automatique afin de contribuer à une meilleure analyse des données. Le contexte de l'IoT sera principalement considéré car les domaines de la gestion et de la sécurité posent certains des plus grands défis dans ce domaine.

Mots clés: Apprentissage automatique, Intelligence artificielle, Analyse des données, IoT, Gestion et sécurité.

خلاصة

تحليل فعال للبيانات بناءً على تقنيات التعلم الآلي، وتطبيقها على إنترنت الأشياء (الإدارة والأمن)

في عالم اليوم، تقوم جميع المؤسسات بتوليد كميات هائلة من البيانات من مصادر متنوعة. سواء كان ذلك من أنظمة المؤسسة نفسها، أو من وسائل التواصل الاجتماعي أو المصادر الأخرى عبر الإنترنت، أو من الهواتف الذكية وأجهزة الحوسبة الطرفية/العميل الأخرى، أو من أجهزة الاستشعار والأدوات التي تشتمل على إنترنت الأشياء، فإن هذه البيانات ذات قيمة كبيرة للمؤسسات التي لديها الأدوات في مكانه للاستفادة منه. يسمى صندوق الأدوات الشامل لهذه الأدوات تحليلات البيانات. وقد يكون هذا الأخير صعبا، نظرا لأنه يتضمن معلومات معقدة ومجموعات بيانات كبيرة يصعب إدارتها يدويا. هذا هو المكان الذي يأتي فيه التعلم الآلي ، فهو يغير الطريقة التي تتعامل بها الشركات القائمة على المستهلك مع البيانات الهائلة التي تولدها. يعتمد تعلم الآلة على الذكاء الاصطناعي على فكرة أن الأنظمة يمكن أن تتعلم تلقائيًا من البيانات، وتحدد الأنماط، وتتخذ القرارات مع القليل من التدخل البشري، ويتم استخدامها لأتمتة بناء النماذج التحليلية. أصبحت البيانات محركًا رئيسيًا للمؤسسات لتعزيز استدامتها في عالم الأعمال التنافسي. ومع إنتاج المزيد من البيانات وتخزينها أكثر من أي وقت مضى، تزايدت أيضًا الحاجة إلى عمليات أكثر كفاءة وفعالية ودقة. يُنظر إلى التعلم الآلي على أنه حل رائع، فهو يقلل بشكل كبير من الجهود، ويوفر الوقت، كما أنه أداة فعالة من حيث التكلفة تحل محل الفرق المتعددة التي تعمل على تحليل ومعالجة وإجراء اختبار الانحدار على البيانات. فهو يعطي نتائج دقيقة ويساعد المؤسسات على بناء نماذج إحصائية تعتمد على البيانات في الوقت الفعلي. الهدف في هذه الأطروحة هو تصميم وتنفيذ حلول فعالة قائمة على التعلم الآلي للمساعدة في إجراء تحليل أفضل للبيانات. سيتم النظر في سياق إنترنت الأشياء بشكل أساسي لأن مجالات الإدارة والأمن تجلب بعضًا من أكبر التحديات في هذا المجال.

الكلمات المفتاحية : التعلم الآلي، الذكاء الاصطناعي، تحليل البيانات، إنترنت الأشياء، الإدارة والأمن.

Contents

List of Figures

List of Tables

List of Abbreviations

AES Advanced Encryption Standard

AI Artificial Intelligence

BGV Brakerski-Gentry-Vaikuntanathan

DES Data Encryption Standard

DGHV Dijk-Gentry-Halevi-Vaikutanathan

ECC Elliptic Curve Cryptography

FHE Fully Homomorphic Encryption

GCD Greatest Common Divisor

HE Homomorphic Encryption

IaaS Infrastructure as a Service

IoT Internet of Things

LFHE Leveled Fully Homomorphic Encryption

LWE Learning With Errors

ML Machine Learning

PaaS Platform as a Service

PHE Partially Homomorphic Encryption

RSA Rivest–Shamir–Adleman

SaaS Software as a Service

SSSP Sparse Subset Sum Problem

SWHE SomeWhat Homomorphic Encryption

VFHE Verifiable Fully Homomorphic Encryption

Introduction

In the digital age, the convergence of Internet of Things (IoT) devices and cloud computing has catalyzed a paradigm shift in data management, ushering in an era of unparalleled connectivity and scalability. This amalgamation offers vast potential for innovation and efficiency across diverse sectors, from healthcare to transportation, agriculture to manufacturing. However, this transformative landscape also brings forth a myriad of challenges, chief among them being the safeguarding of data privacy and integrity amidst the vast expanses of cyberspace.

The proliferation of IoT devices, characterized by their constrained resources and ubiquitous presence, presents a unique set of security concerns. These devices, often operating on the fringes of networks, are prime targets for malicious actors seeking to exploit vulnerabilities and compromise sensitive information. Furthermore, the reliance on cloud infrastructure for data storage and processing introduces additional layers of complexity, as the outsourcing of critical functions necessitates a delicate balance between convenience and security.

In this context, the concept of homomorphic encryption emerges as a beacon of hope, offering a novel approach to data security that transcends traditional encryption methods. Unlike conventional encryption techniques, which require decryption prior to performing operations, homomorphic encryption enables computations to be performed directly on encrypted data, preserving its confidentiality throughout the entire process. This paradigm shift not only bolsters privacy protection but also opens up new avenues for secure computation in distributed environments.

However, the adoption of homomorphic encryption is not without its challenges. The computational overhead associated with homomorphic operations, coupled with the resource constraints of IoT devices, poses significant barriers to widespread implementation. Moreover, ensuring the integrity of homomorphically encrypted data, particularly in cloud-executed operations, remains a pressing concern, as the outsourcing of computation introduces new attack vectors and vulnerabilities.

This dissertation is not directly focused on the application of data analysis or machine learning techniques. Instead, it centers on the management and security of distributed systems and IoT environments, laying the groundwork for efficient machine learning and data analysis in the future. By addressing fundamental challenges in IoT security, encryption efficiency, and data integrity, this research

aims to create a more stable and secure infrastructure upon which advanced data analytics can be applied effectively.

Against this backdrop, this dissertation seeks to explore novel solutions to the myriad challenges facing data security and privacy in an increasingly interconnected world. Through a series of innovative contributions, we aim to address critical gaps in existing encryption schemes, enhance data integrity verification mechanisms, and mitigate the burgeoning size of blockchain data. By leveraging cutting-edge techniques from cryptography, data compression, and distributed systems, we endeavor to pave the way for a more secure and resilient digital ecosystem—one where efficient data analysis and machine learning can be applied seamlessly and effectively in the future.

Main contributions

1. **Lightweight homomorphic encryption scheme**: Our research introduces a novel lightweight asymmetric encryption technique designed specifically for resource-constrained IoT devices. By leveraging a multi-key based approach, we significantly reduce the computational overhead of homomorphic operations while maintaining robust security guarantees. This contribution lays the groundwork for secure and efficient data processing in IoT environments.

2. **Integrity verification for cloud-executed homomorphic operations**: In response to the growing need for data integrity assurance in cloud environments, we propose a simple yet effective checksum mechanism for homomorphically encrypted data. This contribution enables data owners to verify the integrity of computations performed in the cloud without compromising privacy or incurring significant computational overhead.

3. **Efficient aggregate signature scheme for blockchain**: To address the scalability challenges plaguing blockchain networks, we present an enhanced threshold RSA-based aggregate signature scheme. By replacing individual transaction signatures with a compact aggregate signature, we significantly reduce blockchain size and network traffic while preserving security and accountability. This contribution paves the way for the widespread adoption of blockchain technology across diverse applications.

4. **Data compression techniques for blockchain**: Recognizing the exponential growth of blockchain data as a critical bottleneck, we propose a novel data compression technique based on binary matrix representations. By converting data into compact vectors, we achieve substantial reductions in storage requirements while enabling efficient data recovery and integrity

verification. This contribution promises to alleviate the scalability challenges facing blockchain networks and unlock new opportunities for decentralized data management.

5. **Integration of cryptographic techniques in IoT**: Lastly, we explore the integration of cryptographic techniques, including homomorphic encryption and threshold signatures, into IoT ecosystems. By leveraging these advanced cryptographic primitives, we enhance the security and privacy of IoT devices and enable secure communication and computation in resource-constrained environments. This contribution represents a significant step towards realizing the full potential of IoT technologies while ensuring robust protection against emerging threats.

Collectively, these contributions represent a comprehensive effort to advance the state-of-the-art in data security, privacy, and integrity in an increasingly interconnected world. Through innovative research and practical solutions, we seek to empower individuals, organizations, and communities to harness the transformative power of digital technologies while safeguarding their most valuable asset: their data.

Structure of the manuscript

This thesis is organized as follows: Commencing with a comprehensive introduction that contextualizes the research, articulating the problem statement, and outlining the study objectives. In Chapter 1, designated as "Background," we establish a solid foundation by delving into fundamental concepts within distributed systems, IoT, cloud computing, and blockchain. Additionally, this chapter expounds upon essential topics such as cybersecurity, cryptography, and encryption techniques. Building upon this foundational knowledge, Chapter 2 provides a more formal elucidation of definitions and concepts related to homomorphic encryption, crucial for the subsequent chapters.

Chapter 3 undertakes an in-depth exploration of homomorphic encryption in the realms of IoT and cloud computing. This involves presenting the proposed techniques, conducting a thorough performance and security analysis, and detailing the implementation. Subsequently, Chapter 4 discusses blockchain for security concerns. This includes a discussion of proposed protocols, efficiency considerations, analysis of potential cryptanalytic attacks, and the presentation of experimental results.

Finally, the concluding chapter summarizes the key findings, discusses the implications of our research, and suggests avenues for future exploration and improvement. By structuring the thesis in this manner, we aim to provide readers with a logical and comprehensive journey through the research, from foundational concepts to innovative solutions and empirical validation.

List of publications

Journal papers:

1. An enhanced threshold RSA-based aggregate signature scheme to reduce blockchain size, IEEE Access, 2023 [1].

   Authors: Khaled Chait, Abdelkader Laouid, Mostefa Kara, Mohammad Hammoudeh, Omar Aldabbas, and Abdullah T. Al-Essa.

Conference papers:

1. A blind digital watermarking approach using palmprint-embedded local binary patterns and arnold cat map transformations, The 6th International Conference on Pattern Analysis and Intelligent Systems (PAIS2024), 2024 [2].

   Authors: Brahim Ferik, Lakhdar Laimeche, Abdallah Meraoumia, Abdelkader Laouid, Khaled Chait, and Muath AlShaikh.

2. Using transformers to classify arabic dialects on social networks, The 6th International Conference on Pattern Analysis and Intelligent Systems (PAIS2024), 2024 [3].

   Authors: Berhoum Adel, Mohammad Charaf Eddine Meftah, Abdelkader Laouid, Khaled Chait, and Mostefa Kara.

3. One digit checksum for data integrity verification of cloud-executed homomorphic encryption operations, The 7th International Conference on Future Networks & Distributed Systems (ICFNDS2023), 2023 [4].

   Authors: Khaled Chait, Mostefa Kara, Abdelkader Laouid, Mohammad Hammoudeh, and Ahcène Bounceur.

4. A binary matrix-based data representation for data compression in blockchain, The 5th International Conference on Blockchain Computing and Applications (BCCA), 2023 [5].

   Authors: Abdelkader Laouid, Mostefa Kara, Mohammed Al-Khalidi, Khaled Chait, Mohammad Hammoudeh, and Ahmed Aziz.

5. Categorization of digital pathology image using deep learning model, The 7th International Conference on Future Networks & Distributed Systems (ICFNDS2023), 2023 [6].

   Authors: Sana Sahar Guia, Abdelkader Laouid, and Khaled Chait.

6. An adaptive image watermarking based on bellman-ford algorithm, The 7th International Conference on Future Networks & Distributed Systems (ICFNDS2023), 2023 [7].

   Authors: Brahim Ferik, Lakhdar Laimeche, Abdallah Meraoumia, Abdelkader Laouid, Muath AlShaikh, and Khaled Chait.

7. A multi-key based lightweight additive homomorphic encryption scheme, International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP), 2021 [8].

   Authors: Khaled Chait, Abdelkader Laouid, Lamri Laouamer, and Mostefa Kara.

Chapter 1

Background

In recent decades, the world has witnessed a profound and pervasive shift—the digital transformation. This transformation has reshaped nearly every aspect of our lives, from how we communicate and work to how we manage our homes and interact with the world. At its core, digital transformation represents the integration of digital technology into all areas of society, fundamentally altering how we live, work, and interact with information and data.

The rise of digitalization is, perhaps, the most significant technological advancement of our time. It is characterized by the widespread adoption of digital devices, the Internet, and advanced computing systems. Our daily lives are now intimately entwined with smartphones, tablets, laptops, and various smart devices. These devices connect us to the digital world, offering unprecedented access to information and services.

Digital transformation is not driven by a single technology but rather by the convergence of various technological trends. These trends include, but are not limited to:

- Distributed systems: The decentralization of computing resources, allowing for scalability, fault tolerance, and efficient data processing.
- Internet of Things: The proliferation of connected devices, sensors, and actuators that enable the collection and exchange of data in real time.
- Cloud computing: The delivery of computing services, including storage, processing, and analysis, over the Internet.
- Blockchain technology: The development of secure, transparent, and decentralized ledgers that underpin cryptocurrencies and data integrity.
- Machine Learning: The utilization of algorithms and models to enable computers to learn from data, make predictions, and improve over time.

– Cybersecurity: The practice of protecting data, systems, and networks from digital threats and unauthorized access.

– Cryptography: The science of secure communication, providing a foundation for data privacy and integrity.

– Homomorphic encryption: A groundbreaking cryptographic technique that allows data to be processed while it remains encrypted.

As digital technology continues to advance and intertwine with various facets of our lives, it presents both tremendous opportunities and challenges. Digital transformation has led to significant improvements in efficiency, productivity, and innovation. It has enabled new business models, improved healthcare outcomes, and enhanced our ability to address global challenges.

However, it has also introduced new concerns, such as data privacy, cybersecurity threats, and the need for secure and efficient data analysis. This dissertation explores the complex interplay between digital technologies and their applications in the context of IoT, distributed systems, and the critical domains of data management and security.

In the subsequent sections, we delve into the intricacies of each technology and its role in addressing the challenges and opportunities presented by digital transformation, ultimately contributing to a more secure and efficient digital landscape.

1.1 Distributed systems

The concept of distributed systems represents a fundamental shift in the way computing resources are organized, managed, and accessed. A distributed system is a network of interconnected computers that work together as a single cohesive unit, sharing resources and information. Unlike traditional centralized systems, where all processing and data storage occur in a single location, distributed systems disperse these tasks across multiple nodes.

Key characteristics

Key characteristics of distributed systems include:

1. Scalability: One of the defining features of distributed systems is their ability to scale easily. As the load or demand on the system grows, additional resources

can be added to the network without necessitating a complete overhaul of the architecture. This scalability is crucial in accommodating an ever-increasing volume of data and users, a common challenge in the digital transformation era.

2. Fault tolerance: Distributed systems are inherently fault-tolerant. With multiple nodes, even if one or more nodes fail or become unavailable, the system can continue to function. This fault tolerance is essential for ensuring uninterrupted operation in critical applications, such as financial transactions or emergency response systems.

3. Efficient data processing: Distributed systems excel in processing large volumes of data efficiently. The data can be processed in parallel across multiple nodes, which significantly speeds up computations. This capability is central to analyzing vast datasets and conducting real-time data processing, a necessity in the digital transformation landscape.

4. Redundancy: To enhance reliability and fault tolerance, distributed systems often incorporate redundancy. Redundant nodes or data copies are maintained so that in the event of a failure, data and operations can seamlessly shift to a backup source. Redundancy is a critical aspect of data integrity and system resilience.

5. Heterogeneity: Distributed systems are characterized by heterogeneity, as they can consist of various hardware and software components from different vendors. These components need to work together cohesively, which requires robust communication protocols and standards.

6. Decentralization: In contrast to centralized systems, where a single point of control exists, distributed systems are decentralized. Decision-making, data storage, and processing are distributed across the network. This decentralization often leads to improved reliability and efficiency.

7. Geographic distribution: Distributed systems can span vast geographic regions. Nodes may be located in different data centers, regions, or even continents. This geographical distribution allows for redundancy, load balancing, and efficient data access for users in diverse locations.

In the digital transformation era, distributed systems form the backbone of various applications, from cloud computing services to IoT networks. They enable the efficient processing of massive datasets, real-time analytics, and the reliable operation of critical services. However, the distributed nature of these systems also introduces challenges related to data management, security, and synchronization, which we will delve into in subsequent chapters.

Importance in modern computing

In the rapidly evolving landscape of modern computing, distributed systems have risen to prominence due to their ability to address the complex challenges and requirements of today's digital world. Their importance is underscored by several key factors:

1. Handling big data: One of the most critical challenges in modern computing is managing and processing vast volumes of data. The era of Big Data has brought about a need for efficient data storage, analysis, and retrieval. Distributed systems are designed to tackle these challenges by providing scalable and parallel data processing capabilities. This is crucial for applications in fields such as data analytics, machine learning, and scientific research, where large datasets are the norm.

2. Cloud computing: The widespread adoption of cloud computing services relies on distributed systems as the backbone of their infrastructure. Cloud providers leverage distributed architectures to offer scalable, on-demand resources to users worldwide. This enables businesses and individuals to access computing power, storage, and applications without the need to invest in dedicated hardware. Distributed systems ensure that the cloud can accommodate diverse workloads and rapidly adapt to changing demands.

3. Internet of Things (IoT): The proliferation of IoT devices, from smart thermostats and wearable gadgets to industrial sensors and autonomous vehicles, generates an immense volume of data. Distributed systems play a pivotal role in handling the diverse data streams from IoT devices. They provide the necessary infrastructure for real-time data processing, efficient data transmission, and seamless integration with cloud services.

4. E-commerce and online services: E-commerce platforms, social media networks, and online services rely on distributed systems to provide fast and reliable access to users across the globe. By distributing resources and content closer to end-users, these systems ensure low latency and high availability. The ability to seamlessly scale up resources during peak demand, such as online shopping events or viral social media trends, demonstrates the importance of distributed systems in this context.

5. Scientific computing: In scientific fields, such as climate modeling, genomics, and particle physics, researchers require substantial computational power and data processing capabilities. Distributed systems enable researchers to harness the power of supercomputers or distributed computing grids, allowing them to run complex simulations and process large datasets efficiently.

6. Real-time applications: Real-time applications, such as online gaming, video streaming, and financial trading systems, depend on distributed systems to deliver data and services with low latency and high reliability. These systems ensure that data is transmitted and processed in near real-time, providing users with a seamless and responsive experience.

7. Disaster recovery and business continuity: The fault-tolerant and redundant nature of distributed systems makes them a critical component of disaster recovery and business continuity strategies. By distributing data and services across multiple locations, businesses can ensure that even in the face of hardware failures or disasters, their operations can continue with minimal disruption.

In summary, the importance of distributed systems in modern computing cannot be overstated. Their ability to handle big data, support cloud computing, accommodate IoT devices, power online services, enable scientific research, deliver real-time applications, and ensure business continuity makes them a linchpin in the digital era. Nevertheless, with these advantages come challenges related to data consistency, fault tolerance, security, and efficient communication.

Challenges and scalability

While distributed systems offer a myriad of advantages, they are not without their unique challenges. Among these, scalability is a critical aspect.

1. Scalability: It is a defining characteristic of distributed systems. It allows the system to adapt to changing workloads and demands by efficiently accommodating additional resources. However, achieving scalability is a complex task, and it comes in two primary forms:

- Vertical scalability: Also known as scaling up. In vertical scalability, individual nodes within the system are enhanced by increasing their computing power, memory, or storage capacity. This approach is suitable for workloads that can be handled by a single powerful machine. However, it has limitations in terms of cost and practicality.
- Horizontal scalability: Often referred to as scaling out, it involves adding more machines or nodes to the distributed system. This approach is more cost-effective and can be applied to workloads that can be divided into smaller tasks that run in parallel across multiple nodes. Achieving horizontal scalability requires effective load balancing and data partitioning strategies.

Challenges in achieving scalability include maintaining performance as the system grows, ensuring data consistency across distributed nodes, and managing the

communication overhead between nodes. Moreover, identifying the optimal point at which to scale, either vertically or horizontally, is a non-trivial decision that depends on the specific characteristics of the application and workload.

2. Data consistency: One of the key challenges in distributed systems is ensuring data consistency. In a distributed environment, data may be replicated across multiple nodes, and updates or changes to data can occur concurrently from different sources. Maintaining data consistency is crucial to prevent conflicts, ensure accurate results, and guarantee that all nodes have access to the most up-to-date information.

Data consistency mechanisms, such as distributed databases, consensus algorithms, and distributed locking, are used to manage data consistency in distributed systems. These mechanisms introduce trade-offs between consistency, availability, and partition tolerance, as defined by the CAP theorem (Consistency, Availability, Partition tolerance).

3. Fault tolerance: Another significant challenge in distributed systems is fault tolerance. Distributed systems need to be resilient in the face of hardware failures, network issues, or other disruptions. A fault-tolerant system can continue to operate, providing uninterrupted service, even when certain components or nodes fail.

Implementing fault tolerance involves redundancy, replication, and strategies for detecting and recovering from failures. Distributed systems must be designed to handle issues like network partitions, which occur when nodes lose connectivity with one another.

4. Communication overhead: Distributed systems require communication between nodes to exchange data and coordinate tasks. While communication is essential for collaboration, it also introduces overhead and latency. Inefficient communication can lead to performance bottlenecks and negatively impact the system's scalability and responsiveness.

Addressing communication overhead involves optimizing data transmission, reducing the frequency of inter-node communication, and employing efficient communication protocols.

In the subsequent chapters of this dissertation, we delve into these challenges and explore strategies for achieving scalability, data consistency, fault tolerance, and efficient communication in the context of distributed systems. These challenges and their solutions are integral to the successful operation of distributed systems within the digital transformation landscape.

1.2 Internet of Things

IoT represents a revolutionary technological paradigm that has quickly gained prominence in the digital transformation era. It is an ecosystem of interconnected physical devices, sensors, and objects that communicate and share data over the Internet. These "things" can range from everyday items like refrigerators and wearable devices to industrial machinery, smart city infrastructure, and agricultural sensors. The central idea behind IoT is to equip objects with sensors and connectivity, allowing them to collect and exchange data, leading to enhanced functionality and efficiency.

Key components

The IoT ecosystem consists of several fundamental components that work in harmony to enable the seamless flow of data and the functionality of IoT devices. These components are integral to the successful operation of IoT applications and services. Let's delve into these key components:

1. Sensors and actuators: Sensors are the data collection and measurement devices in the IoT ecosystem. They are equipped to monitor and measure various environmental conditions, such as temperature, humidity, light, pressure, and motion. Sensors can also include more specialized devices like GPS modules, biometric sensors, and gas detectors. Their purpose is to collect data from the physical world.

Actuators are the counterparts to sensors. They are responsible for taking actions based on the data received. Common examples of actuators include motors, servos, relays, and solenoids. Actuators allow IoT devices to interact with and influence their surroundings, turning data insights into practical applications. For instance, an IoT thermostat might use a temperature sensor to detect a rise in temperature and activate the air conditioning (actuator) to cool the room.

2. Connectivity: Connectivity is the lifeline of the IoT ecosystem, as it enables devices to communicate and share data. IoT devices rely on a variety of communication technologies to establish connectivity:

- Wi-Fi: Which is commonly used for devices within homes and businesses, providing high-speed internet access. It is ideal for devices that have access to a local Wi-Fi network.
- Cellular networks: Which include 4G and 5G, offer wide-ranging coverage, making them suitable for mobile IoT devices and applications that require connectivity in remote areas.

- Bluetooth: This technology is often used for short-range communications, connecting devices like smartphones and IoT peripherals.
- Low Power Wide Area Network (LPWAN): LPWAN technologies, such as LoRaWAN and Sigfox, are designed for low-power, long-range communication. They are ideal for battery-operated IoT devices that need to transmit data over long distances.
- RFID (Radio-Frequency Identification): RFID technology uses electromagnetic fields to automatically identify and track tags attached to objects. It is widely used for inventory and asset tracking.

3. Cloud services: They play a crucial role in the IoT ecosystem by providing a platform for storing, analyzing, and processing the vast amounts of data generated by IoT devices. One of their key advantages is scalability, as they can handle large data volumes and expand as IoT applications and services grow. Additionally, cloud storage ensures accessibility, allowing users to remotely monitor and control devices from anywhere with an internet connection. Cloud platforms also facilitate data analytics by offering the computational resources needed to extract meaningful insights. Moreover, they provide robust data management solutions that ensure the reliability, availability, and security of IoT data, making them an essential component of modern IoT infrastructures.

4. Data processing: Data processing and analysis are essential for transforming raw IoT data into actionable insights. Real-time data processing allows for immediate responses to incoming information, enabling quick decision-making and automation. Predictive analytics, powered by machine learning and artificial intelligence, helps forecast trends and potential events based on historical data, improving efficiency and preparedness. Through data analysis, systems and processes can be optimized, such as enabling predictive maintenance in industrial settings or enhancing energy efficiency in smart buildings. Additionally, data visualization through dashboards and reports makes complex IoT data more accessible and actionable for organizations and end-users, ensuring informed decision-making.

5. Applications and user interfaces: IoT applications and user interfaces provide end-users with a means to interact with and make sense of the data collected. These applications range from smart home control panels to industrial monitoring dashboards.

These key components form the foundation of the IoT ecosystem, working together to collect, transmit, process, and leverage data from the physical world.

Rapid growth of IoT

IoT's growth has been nothing short of extraordinary. The proliferation of devices is a testament to its importance in today's interconnected world. Several factors have contributed to its rapid expansion:

1. Advancements in connectivity: The availability of affordable and low-power connectivity options has allowed a wide range of devices to become part of the IoT landscape. From 5G networks to LPWAN technologies, these connectivity solutions have made it feasible for even battery-operated devices to join the IoT.

2. Miniaturization and cost reduction: The shrinking size and cost reduction of sensors and hardware components have made it economically viable to integrate them into various products. As a result, IoT capabilities have been incorporated into numerous consumer and industrial devices.

3. Data-driven insights: The value of data-driven decision-making is increasingly recognized across industries. IoT generates a wealth of data that organizations can harness for optimization, predictive maintenance, and improving the user experience.

4. Industrial and enterprise adoption: Enterprises and industries have embraced IoT for monitoring, automation, and process optimization. From logistics and agriculture to manufacturing and healthcare, IoT is enhancing operational efficiency and productivity.

5. Smart cities and infrastructure: Many cities are adopting IoT technologies to improve public services and infrastructure. Smart traffic management, waste collection, and environmental monitoring are just a few examples of how IoT is making cities more efficient and sustainable.

Applications of IoT

The versatility of IoT technology has led to its adoption across a wide range of sectors. In each of these sectors, IoT applications bring unique benefits and opportunities, reshaping industries and enhancing processes. Here are some key sectors where IoT is making a significant impact:

1. Smart homes: IoT has revolutionized the way we interact with our homes. Smart thermostats, lighting systems, security cameras, and voice-activated assistants are just a few examples of IoT devices that enhance comfort, convenience, and energy efficiency for consumers. Smart homes can be remotely monitored and controlled using smartphone apps.

2. Healthcare: In the healthcare sector, IoT devices are instrumental in remote patient monitoring, medication adherence, and the collection of real-time health data. Wearable fitness trackers and medical devices provide doctors and patients with valuable health insights, improving patient care and enabling early intervention.

3. Agriculture: IoT has found applications in precision agriculture, where sensors and data analytics help farmers optimize crop management, reduce water usage, and increase yields. Smart farming techniques, driven by IoT, promote sustainability and resource efficiency.

4. Industrial IoT (IIoT): In the industrial sector, IIoT is revolutionizing manufacturing and logistics. Sensors and actuators in factories and supply chains enable predictive maintenance, process optimization, and the efficient management of assets. These advancements lead to increased productivity and cost savings.

5. Smart cities: Cities around the world are adopting IoT to become smarter and more sustainable. IoT technology is used in traffic management, waste collection, public safety, and environmental monitoring. Smart cities aim to enhance the quality of life for citizens while improving resource management.

6. Energy: IoT is crucial in the energy sector for monitoring and controlling energy consumption. Smart grids, enabled by IoT, improve the efficiency of energy distribution and enable utilities to respond to demand fluctuations. In the renewable energy sector, IoT plays a key role in optimizing the performance of solar and wind farms.

7. Logistics and supply chain: IoT is transforming logistics and supply chain operations. Asset tracking and real-time monitoring of shipments ensure that goods are handled efficiently, reducing losses and improving overall supply chain visibility.

8. Environmental monitoring: Environmental agencies use IoT technology to collect data on air quality, water quality, and weather conditions. Real-time data from sensors helps in assessing environmental changes and taking measures to protect the environment.

9. Retail: Retail businesses use IoT for inventory management, customer engagement, and in-store analytics. Beacon technology and RFID tags enable personalized marketing and improved shopping experiences.

10. Transportation: IoT applications in transportation include real-time tracking of vehicles, predictive maintenance for fleets, and the development of autonomous vehicles. These advancements improve road safety and reduce transportation costs.

11. Education: In the education sector, IoT is used for smart classrooms and campus management. IoT solutions enhance the learning experience and improve security on educational campuses.

12. Public safety: IoT devices like body-worn cameras, gunshot detection systems, and smart traffic management systems are employed by law enforcement agencies to enhance public safety and security.

The adoption of IoT in these sectors underscores its transformative potential. While the applications are diverse, they share common goals of improving efficiency, reducing costs, and enhancing user experiences.

Challenges and considerations

While IoT holds immense promise, it also presents challenges. Issues related to data security, privacy, and the management of vast amounts of data are critical considerations. Additionally, ensuring the interoperability of diverse IoT devices and platforms is an ongoing challenge.

In the chapters that follow, we explore the various aspects of IoT and the vital role it plays in the digital transformation. We also delve into the specific challenges and solutions related to IoT data management and security.

1.3 Cloud computing

Cloud computing is a foundational technology that underpins many aspects of digital transformation. It revolutionizes the way computing resources are provisioned, managed, and accessed. At its core, cloud computing involves the delivery of computing services, such as storage, processing power, and software applications, over the Internet. These services are hosted and managed by cloud service providers, eliminating the need for organizations and individuals to maintain their own physical infrastructure.

Basic concepts

To understand the fundamentals of cloud computing, it is essential to explore its key components. The service models define how cloud resources are delivered. Infrastructure as a Service (IaaS) provides virtualized computing resources over the internet, including virtual machines, storage, and networking, allowing users

to control and manage the operating system and applications. Platform as a Service (PaaS) offers a platform and environment for developers to build, deploy, and manage applications without worrying about the underlying infrastructure, thereby streamlining the development process. Software as a Service (SaaS) delivers software applications over the internet on a subscription basis, allowing users to access them through web browsers without the need for local installations.

Cloud deployment models define how cloud environments are structured. The public cloud is provided by cloud service providers to multiple organizations and users, making resources accessible over the Internet. The private cloud is dedicated to a single organization, often hosted on-premises or by a third-party provider, serving only that organization's needs. A hybrid cloud combines public and private cloud resources, allowing data and applications to be shared between them, thus providing flexibility and scalability.

Cloud computing is characterized by several essential features. On-demand self-service enables users to provision computing resources as needed without requiring human intervention. Broad network access ensures that cloud services can be accessed via various devices, including laptops, smartphones, and tablets. Resource pooling allows computing resources to be shared among multiple users, with dynamic allocation and reassignment based on demand. Rapid elasticity ensures that cloud resources can be quickly scaled up or down to accommodate changing workloads. Additionally, cloud services follow a measured service model, where resource usage is metered and users are billed based on their consumption.

By integrating these key components and characteristics, cloud computing provides a robust and scalable solution that supports modern digital applications, enterprise operations, and personal computing needs.

Advantages of cloud computing

The adoption of cloud computing is driven by a wide array of advantages that cater to the diverse needs of organizations and individuals in the digital transformation era. Here are some of the key advantages of cloud computing:

1. Cost-efficiency: Cloud computing reduces capital expenditure by eliminating the need for organizations to invest in and maintain their own physical hardware and data centers, allowing resources to be redirected to strategic initiatives. The pay-as-you-go model ensures organizations only pay for the resources they use, making it a cost-effective solution. Furthermore, cloud providers benefit from economies of scale, offering resources and services at lower costs than what organizations could achieve in-house.

2. Scalability: Cloud services enable rapid scalability, allowing organizations to quickly adjust resources based on fluctuating workloads or sudden demand surges. Resource pooling ensures efficient allocation, with cloud providers dynamically reallocating resources among users to meet varying demands.

3. Flexibility: Cloud providers offer a diverse range of services, from infrastructure to software applications, allowing organizations to tailor solutions to specific needs. Additionally, remote access ensures that resources are available from any location with internet connectivity, enabling remote work, global accessibility, and increased mobility.

4. Accessibility: With a global reach provided by data centers in multiple regions, cloud services ensure users can access resources from different geographic locations. These services are also device-agnostic, allowing access via laptops, smartphones, tablets, or IoT devices for enhanced convenience.

5. Security and compliance: Reputable cloud providers implement robust security measures, such as encryption, access controls, and threat detection, often surpassing what individual organizations can achieve. Many providers adhere to industry-specific compliance standards, simplifying regulatory requirements for organizations in regulated industries.

6. Disaster recovery: Cloud providers often include data redundancy and backup solutions to safeguard against data loss from hardware failures or disasters. They also offer disaster recovery services, ensuring rapid recovery of data and applications in the event of outages or other emergencies.

7. Agility and innovation: Cloud services accelerate the deployment of applications and resources, reducing the time-to-market for new projects. They provide scalable environments for development and testing, enabling developers to experiment and iterate quickly. Continuous updates from providers allow organizations to access the latest technologies with minimal effort.

8. Environmental benefits: By sharing resources among multiple users, cloud computing optimizes resource utilization, reducing energy consumption and environmental impact. Many providers adopt sustainable practices, such as utilizing renewable energy sources in their data centers, contributing to environmental sustainability.

These advantages have made cloud computing a cornerstone of digital transformation, empowering organizations to streamline operations, reduce costs, enhance security, and foster innovation.

Challenges and security concerns

While cloud computing offers numerous advantages, it also presents challenges and security concerns that organizations and individuals must address:

1. Data security: Storing sensitive data in the cloud introduces risks of data breaches and unauthorized access. Organizations must implement strong encryption for data both in transit and at rest, alongside robust access controls and authentication mechanisms.

2. Downtime and reliability: Cloud services can face outages, leading to interruptions in access to resources. Organizations should understand their provider's Service-Level Agreements (SLAs) and prepare contingencies for outages. Additionally, reliance on internet connectivity can disrupt access to resources during connectivity issues.

3. Data transfer costs: Transmitting data to and from the cloud can incur significant expenses, particularly with large volumes of data. Organizations should assess these costs when planning cloud adoption.

4. Vendor lock-in: Migrating data and applications between cloud providers can be complex and costly, potentially making organizations reliant on specific services and APIs of their chosen provider.

5. Compliance and governance: Organizations in regulated industries must ensure their cloud usage complies with specific regulations and governance requirements. Data sovereignty rules may also dictate where data can be stored and processed, influencing the choice of cloud regions.

6. Resource mismanagement: Over-provisioning resources can lead to unnecessary costs, while under-provisioning may result in performance issues. Effective resource management is crucial to balance costs and performance.

7. Data privacy: Hosting data in shared infrastructure can raise privacy concerns. Organizations must evaluate the implications of shared environments and implement privacy safeguards.

8. Identity and access management: Ensuring proper access control is critical to prevent unauthorized access to cloud resources. Implementing multi-factor authentication enhances security but requires careful management to balance security and usability.

9. Legacy systems integration: Integrating legacy systems with cloud services can be challenging. Organizations must develop strategies to connect on-premises systems with cloud resources effectively.

10. Resource management: Monitoring and controlling cloud costs can be complex, especially in organizations with multiple users or departments. Optimizing resource utilization is an ongoing challenge to ensure cost-efficiency and performance.

Addressing these challenges requires comprehensive strategies that include robust security measures, compliance efforts, and vigilant resource management. By tackling these issues effectively, organizations can maximize the benefits of cloud computing in the digital transformation era.

In the subsequent chapters, we explore strategies for enhancing security in the cloud, managing costs, and optimizing resources to maximize the benefits of cloud computing in the context of digital transformation.

1.4 Blockchain

Blockchain technology has emerged as a transformative innovation, offering decentralized and transparent solutions to various challenges in the digital world. Its unique characteristics, such as immutability, security, and trustlessness, have made it an integral part of modern technology systems. This section provides an overview of blockchain, its core concepts, and its applications in critical areas.

Core concepts

At its core, blockchain is a distributed ledger technology that records transactions across a network of nodes in a secure, transparent, and tamper-proof manner. Each block in the chain contains a list of transactions, a timestamp, and a cryptographic hash of the previous block. This structure ensures that data integrity is maintained and makes it nearly impossible to alter past records without consensus from the entire network.

Blockchain operates through consensus mechanisms, such as Proof of Work (PoW), Proof of Stake (PoS), and Delegated Proof of Stake (DPoS), which validate and secure transactions. These mechanisms ensure trust among participants without requiring centralized intermediaries. Decentralization eliminates single points of failure, reduces operational costs, and provides a resilient infrastructure for data management and transaction processing. Smart contracts, another critical aspect of blockchain, enable automated and self-executing agreements based on predefined rules, further enhancing efficiency and reliability.

Data integrity and transparency

Blockchain's ability to ensure data integrity and transparency has led to its adoption across diverse industries, solving critical issues related to trust and accountability.

1. Supply chain management: Blockchain enables real-time tracking and tracing of goods, improving visibility and authenticity throughout the supply chain. It ensures that each transaction, from raw material sourcing to final delivery, is immutable and accessible to stakeholders.

2. Finance and banking: In the financial sector, blockchain facilitates secure peer-to-peer transactions, cross-border payments, and decentralized finance (DeFi) solutions. Its transparent ledger reduces fraud, minimizes transaction settlement times, and enables access to financial services in underbanked regions.

3. Healthcare: Blockchain ensures secure and interoperable storage of patient records, allowing stakeholders to share critical medical information while maintaining privacy and compliance with regulations such as GDPR and HIPAA. It also enhances traceability in pharmaceutical supply chains, reducing counterfeit drugs.

4. Government and voting systems: Blockchain is being explored for secure and transparent voting systems, ensuring that election results are tamper-proof and verifiable. In public administration, it is used to improve transparency in record-keeping, such as land registries and tax systems.

Integration with IoT

The integration of blockchain with IoT has unlocked new opportunities in secure data exchange and network automation. IoT devices generate vast amounts of data, often processed through centralized servers, creating vulnerabilities such as data breaches and unauthorized access. Blockchain addresses these issues through decentralization and immutable record-keeping.

By providing a decentralized framework for managing IoT devices, blockchain enables secure communication while eliminating reliance on central authorities. Each device in the network can be authenticated and verified through blockchain, reducing the risk of malicious actors compromising the system. Additionally, blockchain ensures that IoT data remains tamper-proof and verifiable, which is particularly important in applications like smart homes, autonomous vehicles, and industrial automation, where data integrity directly impacts safety and efficiency.

Beyond security, blockchain enhances transparency and auditing by creating an immutable audit trail of all interactions between IoT devices. This improves accountability and enables real-time monitoring, which is crucial in critical infrastructure such as smart grids and supply chain automation, where accurate data is essential for operational success.

As blockchain continues to evolve, its integration with emerging technologies like Artificial Intelligence (AI), Machine Learning (ML), and edge computing is expected to drive further innovation. Blockchain can enhance AI's decision-making by providing secure and verified data sources, ensuring more reliable outcomes. Similarly, edge devices can leverage blockchain for decentralized processing and secure interactions without requiring central oversight.

By combining its foundational principles with cutting-edge advancements, blockchain is not only transforming existing systems but also creating new paradigms in data management, security, and transparency. Its role in enhancing trust, efficiency, and accountability positions it as a cornerstone of the digital economy and a critical enabler of technological advancement across industries.

1.5 Cybersecurity and data protection

In the digital age, cybersecurity and data protection have become critical to ensuring the safety and privacy of information systems, particularly as the adoption of IoT and distributed systems continues to grow. This section explores the importance of cybersecurity in IoT, the threats and vulnerabilities facing distributed systems, and key measures for data protection.

Cybersecurity in IoT

The IoT has transformed how devices communicate and share data, enabling innovation in areas such as smart homes, healthcare, transportation, and industrial automation. However, this interconnectivity introduces significant cybersecurity challenges. IoT devices are often deployed in large numbers, operate on varied platforms, and may lack robust security protocols, making them attractive targets for cyberattacks.

Securing IoT ecosystems is essential to protect sensitive data, prevent unauthorized access, and ensure the reliability of critical systems. For example, a cyberattack on connected medical devices could jeopardize patient safety, while a breach in smart grid infrastructure could disrupt energy distribution. Cybersecurity in

IoT not only safeguards data integrity and confidentiality but also builds trust among users and stakeholders, fostering broader adoption of IoT technologies.

Cybersecurity in distributed systems

Distributed systems, characterized by their reliance on interconnected components, are increasingly used for scalable and efficient computing. However, their decentralized nature introduces unique threats and vulnerabilities that must be addressed.

1. Distributed denial-of-service (DDoS) attacks: Attackers can overwhelm distributed systems by flooding them with excessive traffic, rendering services unavailable. These attacks exploit the interconnected nature of distributed systems to amplify their impact.

2. Unauthorized access: Weak authentication mechanisms can allow attackers to gain unauthorized access to nodes or data within distributed systems, leading to data breaches and service disruptions.

3. Data tampering: Without robust security measures, data transmitted across distributed systems may be intercepted and altered, compromising its integrity and reliability.

4. Insider threats: Malicious or negligent actions by authorized users can lead to significant security breaches in distributed systems.

The complexity of distributed systems also increases the difficulty of detecting and mitigating security incidents, underscoring the need for proactive and adaptive cybersecurity measures.

Data protection measures

To combat cybersecurity threats and ensure data protection, organizations must adopt comprehensive strategies that address both technological and procedural aspects of security. One of the fundamental measures is encryption, which safeguards data both in transit and at rest, ensuring that intercepted information remains unreadable to unauthorized parties. Modern encryption algorithms provide robust protection against evolving threats.

Access control mechanisms, such as Role-Based Access Control (RBAC) and Multi-Factor Authentication (MFA), help minimize the risk of unauthorized access to sensitive systems and data. Regularly reviewing and updating access permissions further strengthens security. Additionally, network segmentation plays a

crucial role in limiting the impact of security breaches by dividing networks into smaller, isolated segments, preventing attackers from moving laterally across systems.

Intrusion Detection and Prevention Systems (IDPS) enhance security by identifying and responding to potential threats in real-time, thereby reducing the likelihood of data breaches or service disruptions. Regular updates and patch management ensure that known vulnerabilities are addressed, minimizing the risk of exploitation by attackers.

Beyond technical measures, employee training is essential in fostering a culture of security awareness. Educating staff on cybersecurity best practices, such as recognizing phishing attempts and using strong passwords, significantly enhances an organization's overall security posture. By integrating these protective measures, organizations can effectively safeguard sensitive data and mitigate cybersecurity risks.

Cybersecurity in emerging technologies

As technologies like artificial intelligence (AI), blockchain, and edge computing gain prominence, they present both opportunities and challenges for cybersecurity. For example, AI can enhance threat detection and response but can also be used by attackers to develop sophisticated malware. Similarly, blockchain offers transparency and immutability, but its integration with existing systems requires careful consideration of security implications.

By implementing robust data protection measures and adapting to the evolving threat landscape, organizations can safeguard their digital assets and maintain the trust of their stakeholders. Cybersecurity and data protection are not just technical requirements but strategic imperatives in an increasingly connected and data-driven world.

1.6 Cryptography

Cryptography, a fundamental pillar of secure communication, lies at the heart of modern computer and information security. Derived from the Greek word "kryptos," meaning "hidden," and "logos," meaning "word," cryptography refers to the scientific art of coding and decoding messages to protect their confidentiality, integrity, and authenticity. This section explores the fundamentals of cryptography, its role in data security, and the various cryptographic techniques and algorithms that shape modern cryptosystems.

Fundamentals of cryptography

The field of cryptology, often abbreviated as crypto, encompasses two closely related disciplines: cryptography and cryptanalysis. While cryptography focuses on creating secure communication protocols to encode and decode messages, cryptanalysis involves the study of breaking those protocols, deciphering codes, and exploiting weaknesses in cryptosystems.

Systems designed for encrypting information are commonly known as cryptosystems or ciphersystems, drawing from the term "cipher," derived from the Arabic "sifr," meaning "zero" or "empty." The terms cipher and crypto are often used interchangeably, with some authors favoring "cipher" to sidestep the religious and cultural associations of "crypt."

Cryptographic systems use algorithms to transform plaintext into ciphertext (encryption) and restore ciphertext back into plaintext (decryption). The terms encryption and encipherment, as well as decryption and decipherment, are often used interchangeably. These systems rely on keys—secret values known only to authorized parties—to ensure security. Without the correct key, an adversary cannot decode encrypted information.

Historically, cryptography was used to secure military and diplomatic communications. Today, its scope extends far beyond, underpinning the security of modern digital systems, networks, and online interactions.

Cryptography and data security

Cryptography plays a critical role in maintaining the pillars of data security, including confidentiality, integrity, authenticity, and nonrepudiation. Confidentiality is achieved through encryption, which transforms data into an unreadable format to ensure that only authorized individuals can access the original content. This protection applies to both stored data, such as files on devices, and data in transit, including network communications. Modern cryptographic protocols like SSL/TLS secure internet traffic, while encryption tools safeguard sensitive files and emails.

Ensuring data integrity is another fundamental aspect of cryptography, preventing unauthorized modifications and guaranteeing accuracy and reliability. Hash functions and Message Authentication Codes (MACs) generate unique digital fingerprints for data, allowing any unauthorized alterations to be detected if the fingerprint changes.

Authenticity in secure communication is established through cryptographic methods such as digital signatures, which verify the identity of users and systems.

Public Key Infrastructure (PKI) enables secure authentication by allowing users to validate digital certificates and signed documents, ensuring that messages or transactions originate from a trusted source.

Cryptography also provides nonrepudiation, ensuring that parties cannot deny their involvement in a transaction or communication. Digital signatures, combined with public key cryptography, create undeniable proof of authorship and participation in electronic transactions, such as stock orders or legal agreements. By integrating these cryptographic principles, organizations can safeguard data against unauthorized access, tampering, and forgery, reinforcing overall security in digital communications and storage.

Techniques and algorithms

Modern cryptography employs a wide range of techniques and algorithms to achieve its goals. These methods are designed to be computationally secure and resistant to attacks by malicious actors. Key cryptographic techniques include:

1. Symmetric key encryption: In symmetric encryption, the same key is used for both encryption and decryption. Algorithms like AES (Advanced Encryption Standard) and DES (Data Encryption Standard) are commonly used for fast and efficient encryption of large datasets.

2. Asymmetric key encryption: Also known as public key cryptography, this method uses a pair of keys—a public key for encryption and a private key for decryption. RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Curve Cryptography) are widely used for secure communication and digital signatures.

3. Hash functions: Cryptographic hash functions, such as SHA-256 (Secure Hash Algorithm), generate fixed-length outputs from input data. They are used for verifying data integrity, digital signatures, and password storage.

4. Digital signatures: These are cryptographic proofs of authenticity that ensure messages or documents have not been tampered with and were created by the claimed sender. Digital signatures rely on asymmetric encryption and are crucial for electronic transactions.

5. Hybrid cryptosystems: Many systems combine symmetric and asymmetric cryptography to leverage the advantages of both. For instance, symmetric encryption secures large amounts of data, while asymmetric encryption securely exchanges the symmetric keys.

Cryptographic algorithms must balance security and performance. They are subject to rigorous testing and validation to resist evolving threats, including attacks

from quantum computers in the near future. Quantum-resistant cryptographic algorithms are an emerging area of research aimed at addressing these challenges.

Cryptography in a connected world

As the digital landscape expands, cryptography remains central to securing IoT devices, distributed systems, and emerging technologies. In IoT ecosystems, lightweight cryptographic techniques ensure secure communication between resource-constrained devices. Distributed systems rely on cryptography for secure data exchange, while blockchain technology integrates cryptographic primitives such as hash functions and digital signatures to ensure immutability and authenticity.

By safeguarding the confidentiality, integrity, and authenticity of data, cryptography empowers individuals, organizations, and governments to communicate securely and confidently in a rapidly evolving digital world. Its continuous evolution, driven by new threats and technological advancements, underscores its importance in maintaining trust and security in the information age.

Chapter 2

Homomorphic Encryption Fundamentals

In the ever-evolving landscape of cryptography, one frontier stands as a testament to the ingenuity of cryptographic research and its far-reaching implications for data privacy and security. This chapter delves into the enigma of Homomorphic Encryption (HE), a revolutionary paradigm that defies traditional cryptographic norms by enabling computations on encrypted data without the need for decryption. We will unravel the underlying principles and mechanisms that render HE a reality, exploring its diverse applications.

2.1 Preliminaries

Embarking on the intricate journey of encryption necessitates a robust understanding of the mathematical foundations that underpin cryptographic systems. This section serves as a gateway, introducing key mathematical concepts like groups, rings, and lattices, laying the groundwork for comprehending both symmetric and asymmetric encryption methodologies.

Groups In the context of encryption, group theory provides a formal framework for understanding algebraic structures crucial to cryptography. A group G consists of a set of elements and an operation $*$ that satisfies these properties:

- Closure: $\forall\ a, b \in G, \quad a * b \in G$
- Associativity: $\forall\ a, b, c \in G, \quad (a * b) * c = a * (b * c)$
- Identity: $\exists\ 0 \in G, \quad a * 0 = a$
- Invertibility: $\forall\ a \in G, \exists\ (-a) \in G, \quad a * (-a) = 0$

An abelian group, also known as a commutative group, is one in which the group operation is commutative. For our purposes, we often deal with additive groups, where the operation is addition; these are formally denoted as $(G, +)$. Additionally, understanding cyclic groups and their properties becomes essential. A cyclic group is generated by a single element, and it exhibits unique properties that align with HE mechanisms. Groups play a foundational role in cryptographic protocols, particularly in key generation and manipulation.

Rings Building on the concept of groups, we extend our exploration to rings and fields. A ring $(R, +, \cdot)$ incorporates two operations, addition and multiplication. It introduces new elements of zero and unity, making it a more versatile structure for cryptographic purposes. Ring properties are:

- Closure: $\forall\ a, b \in R, \quad a + b \in R, a \cdot b \in R$
- Commutativity (addition): $\forall\ a, b \in R, \quad a \cdot b = b \cdot a$
- Commutativity (multiplication): $\forall\ a, b, c \in R, \quad (a \cdot b) \cdot c = a \cdot (b \cdot c)$
- Associativity (addition): $\forall\ a, b, c \in R, \quad (a + b) + c = a + (b + c)$
- Associativity (multiplication): $\forall\ a, b, c \in R, \quad (a \cdot b) \cdot c = a \cdot (b \cdot c)$
- Distributivity: $\forall\ a, b, c \in R, \quad a \cdot (b + c) = a \cdot b + a \cdot c$
- Identity (additive): $\forall\ a \in R, \exists\ 0 \in R, \quad a + 0 = a$
- Invertibility (additive): $\forall\ a \in R, \exists\ (-a) \in R, \quad a + (-a) = 0$

Fields Fields elevate our exploration, providing a more refined algebraic structure. A field F is a set equipped with two operations, addition and multiplication, such that it satisfies all the properties of a ring and introduces multiplicative inverses:

- Identity (multiplicative): $\forall\ a \in F, \exists\ 1 \in F \quad a \cdot 1 = a$
- Invertibility (multiplicative): $\forall\ a \in F \setminus \{0\}, \exists\ a^{-1} \in F, \quad a \cdot a^{-1} = 1$

The multiplicative inverses concept ensures that every nonzero element has a reciprocal within the set. This property is crucial for cryptographic protocols, providing a means for division operations: The existence of a multiplicative identity and inverses facilitates the manipulation of encrypted data while preserving the algebraic integrity necessary for secure computations. Fields, therefore, offer a more refined mathematical structure that aligns seamlessly with the requirements of HE protocols.

Lattices Lattices are indispensable in the world of HE, especially in lattice-based cryptographic systems. A lattice is a geometric arrangement of points in a multi-dimensional space. Precisely, a lattice in k-dimensional space is a discrete set of points generated by linear combinations of basis vectors. In other words, it's a discrete additive subgroup of $\mathbb{R}^n$ which can be represented as $\Lambda = \{\gamma_1 \cdot b_1 + \gamma_2 \cdot b_2 + \ldots + \gamma_k \cdot b_k \mid \gamma_i \in \mathbb{Z}\}$, where $B = (b_1, b_2, \ldots, b_k)$ are linearly independent vectors in $\mathbb{R}^n$. The term B is called base of the lattice.

2.2 Modular arithmetic

Modular arithmetic is another cornerstone, especially in algorithms relying on modular operations. Modular arithmetic is a system of arithmetic where numbers "wrap around" after reaching a certain value, known as the modulus. For example, in the modulo 5 system, the numbers 0 through 4 are used, and any number greater than 4 is reduced to its remainder when divided by 5. This means that 6 mod 5 is equal to 1, 7 mod 5 is equal to 2, 8 mod 5 is equal to 3, and so on.

The basic operations of addition, subtraction, and multiplication can be performed in modular arithmetic just like in regular arithmetic, with the difference being that the results are reduced modulo the modulus. For example, in the modulo 7 system, the sum of 3 and 5 is 8, but since 8 is greater than 7, we take its remainder when divided by 7, which is 1. Therefore, $3 + 5 \bmod 7$ is equal to 1.

An essential property of modular arithmetic is that it forms a group under addition, meaning that any two numbers a and b in the modulo n system have a sum $(a + b) \bmod n$ within the modulo n system. This group includes an identity element (0) and inverse elements $(-a)$, such that $a + (-a) \bmod n$ equals 0. Modular multiplication also exhibits intriguing properties. For example, if a and b are two integers that are relatively prime to the modulus n, i.e., their greatest common divisor (GCD) is 1, then their product $a \cdot b \bmod n$ is also relatively prime to n. This implies that there exist integers x and y such that $(a \cdot x + b \cdot y) \bmod n$ is equal to 1, which is known as the extended Euclidean algorithm.

Another important operation in modular arithmetic is exponentiation. The power of a number a raised to an exponent k modulo n ($a^k \bmod n$) can be efficiently computed using algorithms such as square-and-multiply [9]. This is a crucial operation in many cryptographic protocols. Additionally, the modular inverse operation is also important in the field of cryptography, particularly for decryption in asymmetric cryptography. The modular inverse of a, denoted as a^{-1}, is the number x such that $a \cdot x \bmod n = 1$. This operation exists if and only if a and n are coprime.

In number theory, the finite group $\mathbb{Z}_t$, consisting of integers modulo t, forms a subgroup of positive integers modulo t. Alternatively denoted as $\mathbb{Z}/t\mathbb{Z}$, this group can be visualized as the set of integers within the range $(-\frac{t}{2}, \frac{t}{2}]$. For example, when $t = 5$, the group $\mathbb{Z}_5$ comprises the elements $\{0, 1, 2, 3, 4\}$. Moreover, when t is prime, this group forms a finite field denoted as $\mathbb{F}_t$. This is the case of the multiplicative group $\mathbb{Z}_t$, which includes integers modulo t that are coprime to t.

Another important concept in modular arithmetic is the concept of congruence and residue classes. Two integers a and b are said to be congruent modulo n, with $n \neq 0$, if and only if n divides their difference $(a-b)$. In other words, a and b have the same remainder when divided by n. This can be expressed mathematically as:

$$a \equiv b \pmod{n} \iff n \mid (a - b) \tag{2.1}$$

The symbol $\equiv$ represents the congruence relation, while $\pmod{n}$ indicates the modulus. Congruences can be manipulated just like equations. For example, if $a \equiv b \pmod{n}$ and $c \equiv d \pmod{n}$, then we have the following properties:

- Addition: $a + c \equiv b + d \pmod{n}$
- Subtraction: $a - c \equiv b - d \pmod{n}$
- Multiplication: $a \cdot c \equiv b \cdot d \pmod{n}$

Additionally, congruences satisfy certain properties:

- Cancellation: If $a \equiv b \pmod{n}$ and c is relatively prime to n, then $ac \equiv bc \pmod{n}$
- Transitivity: If $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$, then $a \equiv c \pmod{n}$

Residue classes are the distinct sets of integers that share the same remainder when divided by n. When working with congruences, we often group integers into residue classes to study their properties. The residue class of an integer a modulo n is denoted as $[a]_n$ or simply $[a]$. Formally, $[a]_n$ is the set of integers b such that $a \equiv b \pmod{n}$, as shown bellow:

$$[a]_n = \{b \in \mathbb{Z} \mid a \equiv b \pmod{n}\} \tag{2.2}$$

For example, the residue class $[2]_5$ consists of all integers that leave a remainder of 2 when divided by 5, such as $-8, -3, 2, 7, 12, \ldots$

Residue classes enable us to partition the integers into distinct sets based on their remainders modulo n. For any integers $a, b, c \in \mathbb{Z}$, and a positive integer n, the following properties hold within residue classes:

- Addition: $[a]_n + [b]_n = [a + b]_n$
- Subtraction: $[a]_n - [b]_n = [a - b]_n$
- Multiplication: $[a]_n \cdot [b]_n = [a \cdot b]_n$

Moreover, residue classes possess cancellation and transitive properties, similar to congruences. These properties allow us to perform arithmetic operations and simplify expressions involving residue classes.

2.3 Encryption

In the realm of cryptography, an algorithm serves as a set of systematic instructions, guiding data transformation for security purposes. Ciphers, on the other hand, are methods employed within these algorithms to convert plaintext into ciphertext, obscuring the original content. Ciphertext represents the encrypted form of a message, while codes involve symbolic representations for words or phrases. Decrypting or deciphering is the process of reversing encryption, revealing the original message from its encrypted form. Conversely, encryption or enciphering involves transforming plaintext into ciphertext, ensuring confidentiality. Keys play a pivotal role, serving as unique pieces of information that control these cryptographic processes. Lastly, plaintext refers to the human-readable, unencrypted form of a message.

Encryption algorithms can be divided into two categories: symmetric and asymmetric. Symmetric encryption uses a single shared key for encryption and decryption, while asymmetric encryption uses a public key for encryption and a private key for decryption. Symmetric encryption is efficient for bulk data encryption, while asymmetric encryption is useful for secure key exchange and digital signatures. See Figure 2.1 for an illustration of these mechanisms.

2.3.1 Symmetric encryption

Symmetric key cryptography is the bedrock of secure communication. A symmetric key algorithm employs a single key (k) for both encryption (Enc) and decryption (Dec). The process is represented as $c = Enc(k, m)$ and $m = Dec(k, c)$,

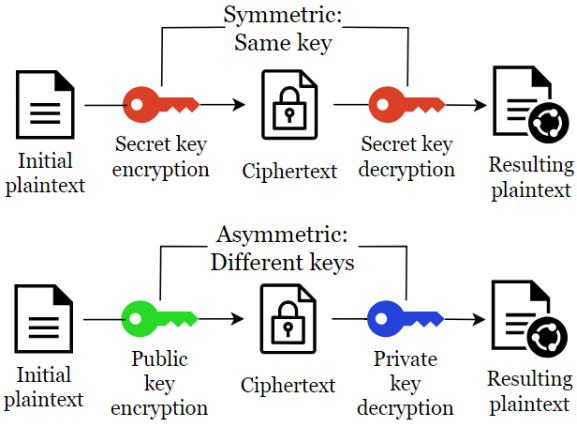

Figure 2.1: Symmetric vs. asymmetric encryption

where c is ciphertext and m is plaintext. Notable symmetric key algorithms include block ciphers (e.g., DES, AES) and stream ciphers (e.g., RC4). The security relies on the key being kept secret.

Delving into modern symmetric encryption, the Advanced Encryption Standard (AES) is a prominent algorithm. It operates on fixed-size blocks and supports key sizes of 128, 192, or 256 bits. The AES encryption process involves substitution (SubBytes), permutation (ShiftRows), mixing (MixColumns), and key addition (AddRoundKey) in multiple rounds. The strength of AES lies in its resistance to various cryptographic attacks.

2.3.2 Asymmetric encryption

Asymmetric key cryptography introduces pairs of keys: a public key pk for encryption and a private key sk for decryption. The security of this system relies on the computational difficulty of deriving the private key from the public key. The encryption process is represented as $c = Enc(pk, m)$, and the decryption process as $m = Dec(sk, c)$. Notable algorithms include RSA and ECC.

RSA (Rivest–Shamir–Adleman) is a widely-used asymmetric algorithm based on the difficulty of factoring large semiprime numbers. Its security relies on the challenge of factoring the product of two large prime numbers. The key generation process involves selecting two large prime numbers, computing their product, and deriving the public and private keys. Elliptic Curve Cryptography (ECC) leverages the mathematical properties of elliptic curves to provide strong security with shorter key lengths compared to RSA. The security is based on the elliptic curve discrete logarithm problem.

2.4 Homomorphic encryption

Homomorphism finds its roots in the Greek words "homos" and "morphe." In Greek, "homos" translates to "same," while "morphe" means "shape" [10]. Consequently, homomorphic can be interpreted as entities possessing a similar shape or form.

In 1978, Rivest, Adleman, and Dertouzos acknowledged the homomorphic features present in different encryption methods, as documented in [11]. Their publication introduced the concept of "privacy homomorphism," suggesting a way to safeguard data while enabling its utilization by untrusted entities. HE actually refers to the algebraic term homomorphism. In algebra, a homomorphism is a structure-preserving map between two algebraic structures. Specifically, if we have two groups $(A, \star)$ and $(B, \bullet)$, a homomorphism φ is a mapping that satisfies

$$\varphi(a \star b) = \varphi(a) \bullet \varphi(b) \tag{2.3}$$

with $a, b \in A$ and $\varphi(a), \varphi(b) \in B$.

This concept can be extended to rings or similar algebraic structures with multiple operations. In the context of cryptography, HE leverages the idea of homomorphisms for secure computations on encrypted data. The mapping, in this case, is represented by the following function

$$Enc : P \to C \tag{2.4}$$

where $(P, +, \cdot)$ is the ring of plaintexts, and $(C, \oplus, \otimes)$ is the ring of ciphertexts. The function Enc is described as a randomized function, meaning it involves a random sampling process from a lattice based on a probability density function. The randomness in the input determines the specific ciphertext a plaintext will be mapped to.

The function Enc is not a deterministic mapping but a randomized one. It takes inputs from the space K, which represents the set of randomized inputs. It is therefore denoted as

$$Enc : P \times K \to C \tag{2.5}$$

indicating that it takes a plaintext and a random input from the key space to produce a ciphertext.

Decryption is intuitively expressed as

$$Dec = Enc^{-1} : C \to P \tag{2.6}$$

Dec is described as a many-to-one function, indicating that multiple ciphertexts can correspond to the same plaintext.

An encryption scheme is considered homomorphic with respect to an operation $\star$ on plaintexts if the decryption of the result of the operation $\bullet$ on the two ciphertexts is equal to the decryption of ciphertext of operation $\star$ performed on the original plaintexts. Mathematically, this is expressed as

$$\begin{aligned} Dec(Enc(m_1) \bullet Enc(m_2)) &= Dec(Enc(m_1 \star m_2)) \\ &= m_1 \star m_2 \end{aligned} \tag{2.7}$$

for some operation $\star$ on plaintexts in P and other operation $\bullet$ on ciphertexts in C.

2.5 HE algorithms

	Key generation ($KeyGen$)	**Encryption** (Enc)	**Decryption** (Dec)	**Evaluation** ($Eval$)
Algo.	$KeyGen(1^K) \rightarrow (pk, pevk, sk)$	$Enc_{pk}(m) \rightarrow c$	$Dec_{sk}(c) \rightarrow m$	$Eval_{pevk}(f, c_1, c_2, \ldots, c_l) \rightarrow c_f$
Input	A security parameter represented as unary, typically denoted as 1^K, which denotes the level of security provided by the encryption scheme	Public encryption key pk and a plaintext single bit message ($m \in \{0, 1\}$)	Secret decryption key sk and ciphertext c	Public evaluation key $pevk$, a set of ciphertexts $\{c_1, c_2, \ldots, c_l\}$, and a function to be applied to the plaintexts
Output	A set formed of a public encryption key pk, public evaluation key $pevk$, and secret decryption key sk	Ciphertext c	Plaintext message m	Ciphertext representing the result of applying the function to the plaintexts
Aim	Generates the necessary keys for encryption, evaluation, and decryption	Encrypts the plaintext message using the public key, resulting in a ciphertext.	Decrypts the ciphertext using the secret key to recover the original plaintext message	Enables computation on encrypted data. It takes a set of ciphertexts and performs a specified operation on the underlying plaintexts without decrypting them

Table 2.1: HE schemes specification

In contrast to encryption schemes employing three algorithms, HE schemes distinguish themselves by incorporating four distinct algorithms. These include the

key generation, encryption, decryption, and additional evaluation algorithms. All these probabilistic polynomial-time (PPT) algorithms form a quadruple that can be expressed as $(KeyGen, Enc, Dec, Eval)$. They are described in table 2.1.

In a HE scheme, the key feature is the ability to perform computations on encrypted data. The encryption algorithm allows for secure transmission of data, the decryption algorithm allows the authorized party to recover the original message, and the evaluation algorithm enables computations on encrypted data without revealing the actual content. This makes it possible to delegate computations to untrusted servers while maintaining the confidentiality of the data.

2.6 Homomorphic scheme types

A cryptographic scheme is deemed "somewhat homomorphic" when its capability to execute operations is constrained by an inability to effectively decrypt beyond a certain threshold of noise introduced during these operations. Specifically, when we label a scheme as additively homomorphic, it implies that an infinite number of operations can be conducted with ciphertexts, each precisely corresponding to an unlimited sequence of additions with plaintexts. Consider another case: a multiplicatively homomorphic scheme should permit an unrestricted number of operations, mirroring the multiplication of plaintexts, prior to the decryption of the resulting ciphertext. A scheme attains full homomorphism when it can seamlessly execute an unlimited number of both addition and multiplication operations. Figure 2.2 summarizes this.

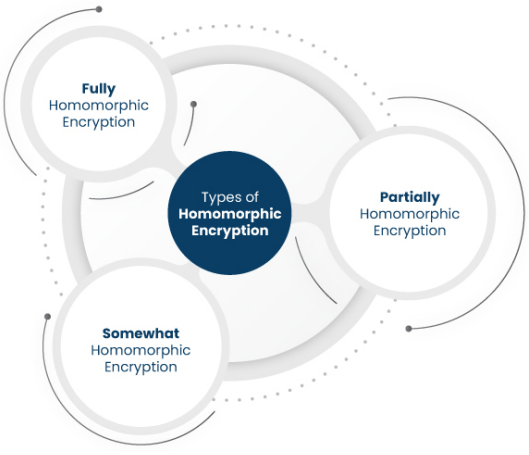

Figure 2.2: Types of HE

2.6.1 Partially homomorphic

PHE schemes stand out based on their ability to homomorphically evaluate specific operations, specifically either addition or multiplication, but not both simultaneously.

2.6.1.1 Addtitive

An additive scheme, which permits an infinite number of operations related to plaintext additions, can be defined as follows:

$$Enc(m_1) \bullet Enc(m_2) = Enc(m_1 + m_2) \tag{2.8}$$

Upon decrypting both sides, the result is the sum of the plaintexts.

Known additive schemes include ElGamal [12], Goldwasser-Micali [13], Benaloh, Paillier [14], and Boneh-Goh-Nissim [15].

To illustrate the concept of additive HE schemes, we explore the additive variant of the ElGamal public key encryption scheme, provided in algorithm 1. To recapitulate, a user generates a public key consisting of parameters (p, α, β) and a private key a. The generator α and the secret exponent a contribute to the public key, while β is derived from $\alpha^a \mod p$. This pairing forms the foundation for secure communication. As randomization is a crucial element in ElGamal, a random value k is generated for each encryption. The sender then computes $x \leftarrow \alpha^k \mod p$ and $y \leftarrow \alpha^m \cdot \beta^k \mod p$, creating the ciphertext (x, y). The randomness in k enhances the security of the encryption. Armed with the private key a, the recipient computes a shared secret using $x^a \mod p$. The original plaintext m is then recovered by calculating $(x^a)^{-1} \mod p$. This process ensures that only the authorized recipient can decipher the message.

Algorithm 1 Additive ElGamal scheme

KEY GENERATION

Require: None

Ensure: public key pk, private key sk

function KEYGEN( )
  $p \leftarrow$ large prime number
  $\alpha \leftarrow$ generator of the multiplicative group modulo p (i.e., $\alpha \in \mathbb{Z}/p\mathbb{Z}^*$)
  $a \leftarrow$ random secret exponent (with $1 < a < p - 1$)
  $\beta \leftarrow \alpha^a \mod p$
  return $pk = (p, \alpha, \beta)$, $sk = a$
end function

ENCRYPTION

Require: public key (p, α, β), plaintext m

Ensure: ciphertext c

function ENC(m)
  $k \leftarrow$ random number (with $1 < k < p - 1$)
  $x \leftarrow \alpha^k \mod p$
  $y \leftarrow \alpha^m \cdot \beta^k \mod p$
  return $c = (x, y)$
end function

DECRYPTION

Require: public key (p, α, β), private key a, ciphertext (x, y)

Ensure: plaintext m

function DEC(c)
  $s \leftarrow x^a \mod p$ (the shared secret)
  $m' \leftarrow y \cdot s^{-1} \mod p$
  recover m from $m' = \alpha^m$
  return m
end function

HOMOMORPHIC ADDITION

Require: public key (p, α, β), ciphertext (x_1, y_1), ciphertext (x_2, y_2)

Ensure: ciphertext $(x_1 \cdot x_2, y_1 \cdot y_2)$

function ADD(c)
  $x_{result} \leftarrow x_1 \cdot x_2 \mod p$
  $y_{result} \leftarrow y_1 \cdot y_2 \mod p$
  return $c_{result} = (x_{result}, y_{result})$
end function

The homomorphism property in ElGamal encryption is manifested by the previously mentioned capability to conduct computations on ciphertexts, aligning with operations performed on the corresponding plaintexts. Specifically, given two plaintexts m_1 and m_2 and their respective ciphertexts $c_1 = Enc(m_1) = (x_1, y_1)$ and $c_2 = Enc(m_2) = (x_2, y_2)$, the homomorphism property allows us to compute an encryption of the sum of the plaintexts $m_1 + m_2$ by executing the componentwise product of the ciphertexts, as shown bellow:

$$\begin{aligned}(x_1 \cdot x_2, y_1 \cdot y_2) &= (\alpha^{k_1} \cdot \alpha^{k_2} \bmod p,\ \alpha^{m_1} \cdot \beta^{k_1} \cdot \alpha^{m_2} \cdot \beta^{k_2} \bmod p) \\ &= (\alpha^{k_1+k_2} \bmod p,\ \alpha^{m_1+m_2} \cdot \beta^{k_1+k_2} \bmod p) \\ &= (\alpha^{k} \bmod p,\ \alpha^{m_1+m_2} \cdot \beta^{k} \bmod p) \\ &= Enc(m_1 + m_2)\end{aligned} \tag{2.9}$$

In order to observe this in practice, let's delve into an illustrative example involving Alice, who seeks to determine the sum of two messages, $m_1 = 15$ and $m_2 = 20$. However, she needs to utilize Bob's calculator for this task. The process unfolds through the additive ElGamal scheme as follows:
- Alice initiates the key pair generation by selecting a prime number $p = 17$. The resulting multiplicative group modulo p is therefore $\mathbb{Z}/_{17}\mathbb{Z}^*$.
- Within this group, she identifies a primitive element, denoted as $\alpha = 4$. Choosing a random secret exponent from the set $\{2, \ldots, 16\}$, she sets $a = 6$, and computes $\beta = 4^6 \mod 17 = 16$. Consequently, Alice's key pair comprises $pk = (17, 4, 16)$ and $sk = 6$.
- Subsequently, she encrypts the messages m_1 and m_2 by selecting corresponding random numbers $k_1 = 7$ and $k_2 = 10$. Using k_1, she calculates $x_1 = 4^7 \mod 17 = 13$ and $y_1 = 4^{15} \cdot 16^7 \mod 17 = 4$. Similarly, employing k_2, she obtains $x_2 = 4^{10} \mod 17 = 16$ and $y_2 = 4^{20} \cdot 16^{10} \mod 17 = 1$.
- Alice concludes by transmitting the ciphertexts $c_1 = (13, 4)$ and $c_2 = (16, 1)$ to Bob.
- Upon receiving Alice's ciphertexts, Bob computes their homomorphic addition: $x_{result} = 13 \cdot 16 \mod 17 = 4$ and $y_{result} = 4 \cdot 1 \mod 17 = 4$. He returns $c_{result} = (4, 4)$ to Alice.
- Alice partially decrypts Bob's response by first computing the shared secret $s = 4^6 \mod 17 = 16$. Subsequently, she calculates $m' = 4 \cdot 16^{-1} \mod 17 = 4 \cdot 16 \mod 17 = 13$. Recognizing that $m' = \alpha^m$, implying $\alpha^m = 13 \mod 17$.
- This result can be verified by examining the initial plaintexts' addition ($m_1 + m_2$): $\alpha^{m_1+m_2} \equiv \alpha^{15+20} \equiv \alpha^{35} \equiv 4^{35} \equiv 13 \pmod{17}$. This aligns precisely with the outcome of the partial decryption process.

Additively homomorphic encryption schemes find considerable utility across various applications. As examples, we highlight three noteworthy implementations. Cohen and Fischer [16] introduced an additively homomorphic encryption scheme

grounded in higher order residuosity, showcasing its efficacy in secure electronic voting, influencing modern web-based voting systems like Helios [17]. Peikert and Waters [18] devised lossy and all-but-one trapdoor functions using additively homomorphic encryption schemes, leveraging these with additional properties to formulate chosen ciphertext secure (CCA-secure) public-key encryption schemes. The third application pertains to Private Information Retrieval (PIR) schemes, which enable users to access information from a database while safeguarding the confidentiality of their queries from the database itself. Kushilevitz and Ostrovsky [19] demonstrated the construction of PIR protocols with sub-linear communication for a single database; this achievement is applicable to any additively homomorphic encryption scheme.

2.6.1.2 Multiplicative

In the same vein of ideas, a multiplicative scheme permits an operation on cyphertexts that is related to plaintexts' multiplication. It can be defined as follows:

$$Enc(m_1) \bullet Enc(m_2) = Enc(m_1 \times m_2) \tag{2.10}$$

Upon decrypting both sides, the end result manifests as the multiplication of the initial plaintexts.

The original ElGamal scheme is an example of a cryptographic system that exhibits multiplicative homomorphism. However, RSA encryption stands out as the most renowned multiplicatively homomorphic scheme. Therefore, it's worth exploring this scheme to get a better understanding of the concept, as detailed in algorithm 2.

RSA cryptosystem [20], proposed in 1978 by Ron Rivest, Adi Shamir, and Leonard Adleman, which uses ideas from number theory, involves choosing two large primes p and q. The public key is formed by their product $n = p \cdot q$, and the private key is derived using Euler's totient function $\varphi(n) = (p-1) \cdot (q-1)$, which determines the number of positive integers smaller than n that are relatively prime to n. The public key comprises n and a carefully chosen e, while the private key involves $d = e^{-1} \mod n$. Encrypting a plaintext m in RSA involves computing $c = m^e \mod n$. The modular arithmetic employed in encryption ensures the confidentiality of the message. To recover the original message, the recipient uses the private key d to calculate $m = c^d \mod n$. The mathematical properties of modular exponentiation guarantee successful decryption, which is effectively executed as $c^d = (m^e)^d \mod n = (m^e)^{e^{-1}} \mod n = m \mod n$.

Algorithm 2 RSA scheme

KEY GENERATION

Require: None

Ensure: public key pk, private key sk

function KEYGEN( )
  $p \leftarrow$ large prime number
  $q \leftarrow$ large prime number
  $n \leftarrow p \cdot q$
  $\varphi(n) \leftarrow (p-1) \cdot (q-1)$
  $e \leftarrow$ random number (with $gcd(e, \varphi(n)) = 1$ and $1 < e < \varphi(n)$)
  $d \leftarrow e^{-1} \mod \varphi(n)$
  return $pk = (n, e)$, $sk = (n, d)$
end function

ENCRYPTION

Require: public key (n, e), plaintext m

Ensure: ciphertext c

function ENC(m)
  $c \leftarrow m^e \mod n$
  return c
end function

DECRYPTION

Require: private key (n, d), ciphertext c

Ensure: plaintext m

function DEC(c)
  $m \leftarrow c^d \mod n$
  return m
end function

HOMOMORPHIC MULTIPLICATION

Require: public key (n, a), ciphertext c_1 and c_2

Ensure: ciphertext $c_1 \cdot c_2$

function MULT(c)
  $c_{result} \leftarrow c_1 \cdot c_2 \mod n$
  return c_{result}
end function

The magic of RSA lies in its homomorphic multiplication capability. Combining two ciphertexts c_1 and c_2 results in a new ciphertext $c_1 \cdot c_2$, encrypting the product

of the respective plaintexts m_1 and m_2, as follows:

$$\begin{aligned} c_1 \cdot c_2 &= m_1^e \cdot m_2^e \mod n \\ &= (m_1 \cdot m_2)^e \mod n \\ &= Enc(m_1 \cdot m_2) \end{aligned} \tag{2.11}$$

To gain a practical insight into the RSA scheme, let's revisit our earlier scenario where Alice enlists Bob's computational assistance without disclosing her sensitive plaintexts. To reiterate, the specific values at play are $m_1 = 15$ and $m_2 = 20$. Our current aim is to perform a secure computation, specifically the multiplication of these two values.
- Alice initiates the key pair generation by selecting two prime numbers $p = 17$ and $p = 11$, so $n = 17 \cdot 11 = 187$.
- Alice initiates the key pair generation by carefully choosing two prime numbers $p = 17$ and $p = 11$, resulting in the composite modulus $n = 17 \cdot 11 = 187$. She then calculates the totient function, denoted as $\varphi = (p1) \cdot (q1) = 1610 = 160$.
- Alice then chooses a public exponent e such that $1 < e < \varphi(n)$ and e is coprime to $\varphi(n)$. Let's say she picks $e = 7$. She calculates her private exponent d as the modular multiplicative inverse of e modulo $\varphi(n)$. So, $d = 23$ since $7 \cdot 23 \equiv 1 \mod 160$. Alice's public key is $(n, e) = (187, 7)$, and her private key is $(n, d) = (187, 23)$.
- Next, Alice encrypts her messages m_1 and m_2 by computing $c_1 \equiv m_1^e \mod n$ and $c_2 \equiv m_2^e \mod n$. Using $e = 7$, she gets $c_1 \equiv 15^7 \mod 187 = 93$ and $c_2 \equiv 20^7 \mod 187 = 147$. She then transmits c_1 and c_2 to Bob.
- Upon receiving the ciphertexts c_1 and c_2, Bob performs the homomorphic multiplication operation: $c_{result} = c_1 \cdot c_2 \mod n$. In this case, $c_{result} \equiv 93 \cdot 147 \mod 187 = 20$. Bob sends c_{result} back to Alice.
- To retrieve the plaintext result, Alice computes $m_{result} \equiv c_{result}^d \mod n$. Using $d = 23$, she gets $m_{result} \equiv 20^{23} \mod 187 = 113$. Therefore, the product of m_1 and m_2 is 113.
- This result aligns with the standard multiplication of the plaintexts: $m_1 \cdot m_2 = 15 \cdot 20 = 300$. The congruence $m_1 \cdot m_2 \equiv 300 \mod 187$ is consistent with the RSA decryption process, confirming the successful homomorphic multiplication using the RSA scheme.

2.6.2 Fully homomorphic

Fully homomorphic encryption (FHE) is characterized by its ability to execute addition and multiplication operations simultaneously, enabling it to compute a wide range of operations. In 2009, Craig Gentry introduced the initial construction of a FHE [21], where he leverages lattice-based cryptography.

2.6.2.1 Lattice-based cryptography

Lattice-based cryptography [22], a type of cryptographic scheme, derives its security from the complexity of certain mathematical problems associated with lattice structures. The underlying mathematical problems that lattice-based cryptography relies on, such as finding the shortest vector in a lattice or approximating the closest vector problem, are considered difficult to solve. One significant advantage of lattice-based cryptography is its resilience to attacks by quantum computers. Unlike some traditional cryptographic schemes, which rely on the challenge of factoring large numbers, lattice-based cryptography remains secure against quantum computers using Shor's algorithm [23]. Consequently, lattice-based cryptography is considered a promising candidate for post-quantum cryptography, offering a robust level of security even as advancements in quantum computing continue.

An example of a lattice-based cryptographic primitive is the Learning With Errors (LWE) [24] problem, where the challenge is distinguishing between true random noise and intentionally introducing noise. LWE is a foundation for building various cryptographic protocols, including public-key encryption, digital signatures, and key exchange.

2.6.2.2 Gentry's scheme

Gentry introduces a comprehensive "blueprint" method consisting of three main components: a Somewhat Homomorphic Encryption (SWHE) scheme, a bootstrapping method to transform SWHE into FHE, and a specialized technique to connect these components.

Gentry's plan for achieving FHE was detailed in his influential work and relied on ideal lattices, with the security of the scheme based on the assumed difficulty of the "ideal coset problem" within these lattices. Gentry also presented an alternative construction based on integer arithmetic in his PhD thesis [25], initially credited to van Dijk. This construction was later completed and analyzed by van Dijk, Gentry, Halevi, and Vaikuntanathan (DGHV) in [26].

DGHV construction The first step is to create a SWHE scheme capable of evaluating low-degree polynomials homomorphically. The problem involves polynomials of the form $p \cdot q_i + r_i$, where p is a random odd number, q_i are chosen from a distribution D which can potentially depend on p, and r_i are small noise terms. The goal is to find the hidden number p. The SWHE encryption scheme works by key generation, encryption, evaluation, and decryption.

To better understand the concepts, we will explore the integer-based scheme proposed by van Dijk et al., which is considered a secret-key encryption scheme. In this scheme, the odd η-bit integer p serves as the secret key ($p \in [2^{\eta-1}, 2^{\eta})$). For a plaintext m, where $m \in \{0, 1\}$, the resulting ciphertext is obtained by adding a large random multiple of p, represented as q, to a randomly chosen integer M. The selection of M ensures that its parity aligns with the value of m, meaning M is even when m equals 0 and odd when m equals 1. The encryption process is defined by the equation:

$$c = m + 2 \cdot r + p \cdot q \tag{2.12}$$

where r is a random integer and $r < \frac{p}{4}$ (or $2 \cdot r < \frac{p}{2}$).

Decryption in this scheme involves reducing the ciphertext modulo p, which represents the "noise parameter", typically falling within the range of $[-\frac{p}{2}, \frac{p}{2})$. The output of the decryption process is the parity of the resulting value, which can be expressed as:

$$m = (c \bmod p) \bmod 2 \tag{2.13}$$

To ensure successful decryption, the noise has to be small enough. More precisely, decryption works as long as $m + 2 \cdot r < \frac{p}{2}$.

When performing addition or multiplication operations on two ciphertexts, denoted as c_1 and c_2, the resulting ciphertexts (c^+ and $c^\times$) maintain a structure similar to the original ones. Specifically:

$$\begin{aligned} c^+ &= c_1 + c_2 \\ &= p \cdot q_1 + 2 \cdot r_1 + m_1 + p \cdot q_2 + 2 \cdot r_2 + m_2 \\ &= p \cdot (q_1 + q_2) + 2 \cdot (r_1 + r_2) + (m_1 + m_2) \\ &= p \cdot q' + 2 \cdot r' + (m_1 \oplus m_2) \end{aligned} \tag{2.14}$$

$$\begin{aligned} c^\times &= c_1 \cdot c_2 \\ &= (p \cdot q_1 + 2 \cdot r_1 + m_1) \cdot (p \cdot q_2 + 2 \cdot r_2 + m_2) \\ &= p \cdot (q_1 \cdot c_2 + c_1 \cdot q_2) + 2 \cdot (m_1 \cdot r_2 + r_1 \cdot m_2 + 2 \cdot r_1 \cdot r_2) + (m_1 \cdot m_2) \\ &= p \cdot q'' + 2 \cdot r'' + (m_1 \cdot m_2) \end{aligned} \tag{2.15}$$

When the initial noise quantities, r_1 and r_2, are sufficiently small compared to p, the resulting ciphertexts c^+ and $c^\times$ can still be decrypted correctly. In other words, if the noise is kept within certain bounds, the decryption process will yield the intended values. Moreover, this scheme allows for the computation of low-degree polynomials on ciphertexts, with the decrypted result corresponding to the evaluation of the same polynomials (over $\mathbb{Z}_2$) applied to the plaintext bits.

The security of this scheme has been proven in [26] under the assumption that the approximate greatest common divisor (approximate-GCD) problem is hard, which,

in essence, involves finding the value of p. It is believed that the approximate-GCD problem is indeed difficult for suitable parameters, further reinforcing the security guarantees of the scheme. However, the described scheme lacks compactness, as the bit size of the ciphertext grows with the degree of the evaluated polynomial. To address this issue, van Dijk et al. proposed a solution involving the publication of multiple near multiples of p with various sizes and using modular reduction. This approach helps to mitigate the growth in ciphertext size. Nonetheless, the scheme faces a more significant challenge related to its limited homomorphic capacity, which is restricted to low-degree polynomials. This limitation arises due to the rapid growth of noise magnitude with the degree of the polynomial, leading to decryption failure when the noise surpasses $\frac{p}{2}$. This noise development problem is inherent in Gentry's construction from [21] and remains a prevalent challenge in every FHE construction developed thus far.

Bootstrapping theorem Bootstrapping is a technique used to reduce the noise in ciphertexts that accumulates during homomorphic operations. This technique, first introduced by Gentry, transforms an SWHE scheme into a leveled FHE scheme. The basic idea is to apply the decryption function homomorphically to a ciphertext, using an encrypted version of the secret key as an input. This process reduces the noise in the ciphertext, allowing for further homomorphic operations to be performed. The resulting scheme is called a bootstrappable encryption scheme.

Bootstrapping can be applied to any ciphertext, enabling the homomorphic evaluation of functions with arbitrarily large depth. However, the decryption circuit must be bootstrappable, meaning that it can be evaluated homomorphically using the bootstrapping procedure.

Let's consider an encryption scheme with two pairs of public and secret keys (pk_1, sk_1) and (pk_2, sk_2), and an encryption algorithm Enc. Let $c = Enc_{pk_1}(m)$ be a ciphertext that encrypts a message m under the public key pk_1. The bootstrapping procedure is performed in three steps:

1. Encrypting the secret key: The first step is to encrypt the secret key sk_1 under a fresh public key pk_2, resulting in an encrypted secret key $sk_1' = Enc_{pk_2}(sk_1)$.

2. Encrypting the ciphertext: The second step is to encrypt the ciphertext c under the fresh public key pk_2, resulting in an encrypted ciphertext $c' = Enc_{pk_2}(c)$.

3. Homomorphic decryption: The third step is to apply the decryption function homomorphically to the encrypted ciphertext c', using the encrypted secret key sk_1' as an input. This results in a new ciphertext c'' that encrypts the same message as the original ciphertext c but with reduced noise.

The homomorphic decryption step can be expressed using the following equation:

$$c'' = Eval(Dec, c', sk_1') \tag{2.16}$$

where $Eval$ is the homomorphic evaluation function that takes as input a function f (in this case, the decryption function Dec), and two ciphertexts c_1 and c_2, and outputs a new ciphertext that encrypts the result of applying f to the plaintexts corresponding to c_1 and c_2. According to this definition, the preceding expression can be further developed as follows:

$$\begin{aligned} Eval(Dec, c', sk_1') &= Dec_{[Enc_{pk_2}(sk_1)]}(Enc_{pk_2}(c)) \\ &= Enc'_{pk_2}(Dec_{sk_1}(c)) \\ &= Enc'_{pk_2}(m) \\ &\approx Enc_{pk_2}(m) \end{aligned} \tag{2.17}$$

The value $Enc'_{pk_2}(m)$ is equivalent to $Enc_{pk_2}(m)$ because both decrypt to m, i.e., $Dec_{sk_2}(Enc'_{pk_2}(m)) = Dec_{sk_2}(Enc_{pk_2}(m)) = m$.

The objective of bootstrapping is to reduce the noise of the ciphertext, which enables further homomorphic operations to be performed. The new ciphertext obtained after bootstrapping will have a higher noise level than a fresh ciphertext but lower than a ciphertext obtained after homomorphically evaluating functions with higher depth than the decryption circuit. The bootstrapping procedure can be applied multiple times to further reduce the noise in the ciphertext. However, each application of bootstrapping increases the computational complexity and memory requirements of the FHE scheme.

The cornerstone result obtained by Gentry is that it suffices to construct a scheme that is capable of evaluating only a particular set of small degree functions, known as the decryption circuit, in order to construct a FHE scheme. If the scheme is secure when the encryption of the secret key is published, then bootstrapping can be applied to obtain a FHE scheme. The bootstrapping theorem states that if an SWHE scheme is bootstrappable and the decryption circuit can be evaluated homomorphically, then the scheme can be transformed into a FHE scheme. The theorem can be expressed using the following equation:

$$Eval(f, Enc_{pk}(m_1), \ldots, Enc_{pk}(m_k)) = Enc_{pk}(f(m_1, \ldots, m_k)) \tag{2.18}$$

where $Eval$ is the homomorphic evaluation function, f is an arbitrary function, and $m_1, \ldots, m_k$ are plaintext messages.

Squashing As mentioned earlier, bootstrapping is a technique used in FHE to reduce the noise in ciphertexts that accumulates during homomorphic operations.

However, Gentry's original bootstrapping procedure was limited to decryption algorithms with small depth, meaning that it could not be applied to more complex functions. To overcome this limitation, Gentry introduced the technique of squashing.

This final step addresses the challenge of applying the bootstrapping theorem to existing SWHE schemes. Some of them, including the approximate-GCD based scheme, have decryption circuits with degrees larger than their homomorphic capacity. Squashing works by selecting a set of vectors $v_1, v_2, \ldots, v_k$ such that their sum equals the multiplicative inverse of the secret key, i.e., $(B^{sk}J)^{-1} = \sum_{i=1}^{k} v_i$. Where B is the base used in the scheme, sk is the secret key, and J is a fixed matrix used in the encoding of the message.

If the ciphertext c, obtained by encrypting m using the public key pk as $Enc_{pk}(m)$, is multiplied by the elements of the set of vectors, the polynomial degree of the circuit is reduced to a manageable level for the scheme. The bootstrapped ciphertext c' can therefore be obtained as follows:

$$c' = Enc_{pk}(mB^z) \cdot \prod_{i=1}^{k} (Enc_{pk}(v_i) + e_i) \pmod{q} \tag{2.19}$$

Here, z is a random integer that is added to the message before encryption to prevent decryption attacks, e_i's are small error terms introduced during the homomorphic multiplication, and q is the modulus used in the scheme. The bootstrapped ciphertext c' now has a smaller noise level compared to the original ciphertext c, allowing for further homomorphic operations to be performed.

The security of the squashing technique is based on the assumption of the Sparse Subset Sum Problem (SSSP). The SSSP is a computational problem in which, given a set of integers and a target sum, the goal is to find a subset of the integers that add up to the target sum. The assumption is that it is computationally hard to solve the SSSP for a randomly chosen set of integers. In the context of squashing, the SSSP assumption is used to prove that it is hard to recover the secret key from the ciphertext and the set of vectors used in the squashing process.

Despite the blueprint's elegance, schemes constructed using it have limitations, including reliance on non-standard cryptographic assumptions and efficiency concerns.

2.7 Conclusion

Homomorphic encryption (HE) represents a groundbreaking advancement in cryptography, enabling computations on encrypted data without decryption. This

chapter explored the mathematical foundations essential to understanding HE, including group theory, ring structures, and modular arithmetic, as well as the core encryption mechanisms that define modern cryptosystems.

We examined different HE schemes, from Partially Homomorphic Encryption (PHE) supporting either addition or multiplication, to Fully Homomorphic Encryption (FHE), which enables arbitrary computations on ciphertexts. Notably, Gentry's breakthrough construction introduced the concept of bootstrapping, addressing noise accumulation and making FHE practically viable.

Despite its immense potential, HE remains computationally expensive, posing challenges in efficiency and implementation. However, ongoing research, particularly in lattice-based cryptography and post-quantum security, continues to refine HE protocols, paving the way for their application in secure cloud computing, privacy-preserving machine learning, and confidential data processing. As the need for secure and private computation grows, HE is poised to play a crucial role in shaping the future of encrypted data processing.

Chapter 3

HE for IoT and Cloud Computing

This chapter discusses the articles: "A multi-key based lightweight additive homomorphic encryption scheme, The International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP'21), El Oued, Algeria, 2021." and "One digit checksum for data integrity verification of cloud-executed homomorphic encryption operations, The International Conference on Future Networks and Distributed Systems (ICFNDS'23), Dubai, UAE, 2023.".

In the subsequent sections, we delve into the details of our proposed partially homomorphic lightweight asymmetric encryption technique designed for IoT devices. Simultaneously, we introduce a lightweight method to verify the integrity of homomorphically encrypted results performed in the cloud. Through a comprehensive examination of our contributions, we aim to provide insights into enhancing the security and functionality of IoT devices operating in cloud environments, addressing the evolving challenges at the intersection of HE and cloud computing.

3.1 Introduction

The rapid proliferation of IoT devices, characterized by their inherent constraints such as low power and limited computational resources, has necessitated innovative solutions to enhance their functionality and secure data management. One promising approach involves the integration of IoT with cloud computing, offering benefits like data scalability, increased device capacity, and economies of scale. However, traditional cloud-based solutions pose privacy risks, mainly when relying on untrustworthy service providers. This has led to the investigation of cutting-edge cryptographic techniques to safeguard sensitive information in IoT environments.

One noteworthy cryptographic advancement in this regard is homomorphic encryption (HE), which, let us recall, is a technique that enables computations to be

performed on encrypted data without decrypting it. HE introduces challenges related to data integrity, especially in cloud-based environments. The vulnerability of homomorphically encrypted data to tampering attacks has prompted ongoing research to develop robust methods for ensuring the integrity of results obtained through homomorphic operations.

In this context, our research delves into the crossroads of IoT, cloud computing, and HE, proposing novel techniques to address critical challenges in the field. We introduce a partially homomorphic lightweight asymmetric encryption technique tailored for IoT devices with limited resources. This technique stands out by allowing the encryption of decimal digits of a message separately, each multiplied by a distinct key. Unlike many existing encryption techniques, our proposed method minimizes ciphertext expansion, keeping it close to the size of the original data. Furthermore, the time complexity of our scheme is optimized to $O(n)$, making it particularly suitable for resource-constrained devices and networks while ensuring high-security standards.

Building on the foundation of HE, we then extend our contributions to address the critical issue of data integrity in cloud-based homomorphic computations. We present a lightweight technique that empowers data owners to verify the integrity of homomorphically encrypted results stored in the cloud. This method introduces a simple yet effective approach by adding a single digit to the encrypted message before storage, serving as verification proof. This digit subsequently enables the validation of HE, ensuring the integrity of the computation results in an untrusted cloud environment. Importantly, our technique is versatile and can be seamlessly integrated with any homomorphic scheme employing encryption with non-isolated plaintext.

The following sections delve into the details of our proposed encryption technique and the method for ensuring verifiable HE. We discuss the motivation, design considerations, and the potential impact of our contributions on enhancing the security and functionality of IoT devices operating in cloud environments. Through rigorous analysis and experimental validation, we demonstrate the effectiveness and efficiency of our proposed techniques, positioning them as valuable contributions to the evolving landscape of secure and privacy-preserving IoT and cloud computing ecosystems.

3.2 Background

Data security is a paramount concern for data owners across various sectors. HE emerges as an up-and-coming solution to guarantee privacy in scenarios where sensitive data requires protection. This cryptographic technique enables adequate

data security, allowing operations to be conducted on encrypted data without compromising its confidentiality. Applications of HE extend to securing data stored in the cloud, providing a means to perform operations on encrypted data, and even facilitating the operation of peer-to-peer cryptocurrencies without the need for a central authority [27].

Encryption, a fundamental aspect of securing data, comes in two commonly used types: symmetric and asymmetric encryption. Symmetric encryption relies on a shared key for both encryption and decryption, while asymmetric encryption involves the distribution of a public key for encryption and a private key for decryption. However, the vulnerability of symmetric encryption lies in the fact that if an adversary gains access to the secret key, they can decrypt all messages encrypted with that key. In contrast, asymmetric encryption introduces a more distributed approach, limiting the potential damage an adversary can cause by obtaining the private key of a specific party.

HE schemes offer varying degrees of computational flexibility, they are categorized into three types: Partially Homomorphic (PHE), Leveled Fully Homomorphic (LFHE), and Fully Homomorphic (FHE). PHE schemes allow one operation type (either addition or multiplication) with an unlimited number of repetitions on encrypted messages [20, 12], while LFHE schemes cover both addition and multiplication but with a limited number of operations, primarily from the circuit depth perspective. The groundbreaking Fully FHE scheme, introduced by Gentry in 2009 [21], enables both addition and multiplication with an unlimited number of operations. This empowers users to perform any arbitrary computation on encrypted data, which unlocks a vast potential for data analysis in the cloud. However, existing FHE schemes often come at a significant cost. The sheer size of keys, ciphertexts, and the computational complexity associated with FHE operations render them impractical for resource-constrained IoT devices. For instance, the Gentry and Halevi FHE scheme requires a public key of 70 megabytes and substantial time for basic operations. To address these limitations, we've proposed an asymmetric Multi-Keys Partially Homomorphic Encryption (MK-PHE) scheme designed for integers, taking into account the feasibility in an IoT environment.

In parallel, cloud computing has transformed the data storage and processing landscape. It operates as an infrastructure where remote servers, accessed through the Internet, manage storage and processing. The flexibility and on-demand nature of cloud services, encompassing Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), provide clients with the necessary tools and resources to meet their diverse needs. Despite the advantages of cloud computing, security and privacy challenges hinder its widespread adoption.

The benefits of cloud services, such as saving storage space and computing time, are countered by significant security and privacy challenges. Clients often lack visibility into how their data is processed, introducing concerns about data control

and privacy. Integrity checking and encryption emerge as crucial measures to protect user privacy and data in the cloud. While data may be encrypted during storage and transmission, the need for decryption by a third party to process data poses serious security risks.

With HE, the cloud can directly operate on encrypted data and return only an encrypted result to the client, mitigating the need for potentially risky data exposure. However, challenges such as feasibility, limitations on ad-hoc queries, the potential for information leakage, noise tracking, and data integrity issues persist in HE, forming the focus of further exploration in our study.

3.3 Related works

In the realm of data security, various encryption schemes have been proposed in the literature to address the growing concerns of data owners. For confidentiality in multi-key environments, Pang et al. [28] introduced an additive homomorphic technique with a double decryption mechanism. Additionally, an innovative lightweight block cipher based on a leveled fully homomorphic cipher scheme was presented by authors in [29], utilizing NTRU as the foundation.

FHE has become a cornerstone in supporting data computation in ciphertext form by untrusted third parties. However, ensuring data integrity in homomorphically encrypted operations is crucial. Various verifiable encryption-decryption schemes have been proposed to address this concern. Viand et al. [30] conducted a comprehensive analysis of existing FHE integrity techniques and proposed a new vision for Verifiable Fully Homomorphic Encryption (VFHE). Huang et al. [31] presented a VFHE scheme that transforms plaintext into a triangular matrix, leveraging matrix blinding technology for privacy protection. However, this scheme requires clients to maintain proof of verification for each operation, imposing constraints on size and complexity.

In [32], El-Yahyaoui et al. proposed a noise-free VFHE technique using Lipschitz's quaternions, allowing computations over encrypted data under a symmetric key. In contrast, Fiore et al. [33] focused on enhancing verifiable delegation of computation on encrypted data. They introduced refined definitions to withstand adversaries and a novel homomorphic hashing technique for efficient computation authentication.

Jin et al. [34] provided a universal construction of FHE and constructed a general VFHE, with the verifiability of an evaluation function as the main objective. Luo et al. [35] introduced a method for verifiable decryption for the Brakerski-Gentry-Vaikuntanathan (BGV) scheme [36], incorporating an interactive zero-knowledge

proof protocol. Awadallah et al. [37] proposed a verification scheme based on modular residue to validate HE computations over an integer finite field.

In this context, our research builds upon these insights to introduce a novel and lightweight technique. This technique incorporates a single-digit checksum into the encrypted message stored in the cloud, providing a swift and secure means for data owners to verify the integrity of homomorphically encrypted results. In the subsequent sections, we present our proposed solution, analyze its robustness, showcase time execution results, and conclude by discussing the broader implications of our work.

3.4 Proposed techniques

3.4.1 HE scheme

The core idea of this technique is to treat the plaintext message as a sequence of decimal digits, where each digit is encrypted separately with a unique key. Specifically, instead of encrypting the entire message, we break it down into individual components, allowing for better efficiency and security in an IoT environment.

Against an untrusted cloud, the user tries to get the result of $\alpha o \beta$. For this, we increase the security level of the technique represented in Equation 3.1, where ψ is the ciphertext, π is the original message, k denotes the private key, and r is a random number.

$$\psi = (\pi \times k + r \times p) \mod n \tag{3.1}$$

The proposed technique can be defined as follows:

Key Generation (KeyGen) For each digit in the plaintext message, a unique private key and public key are generated. This ensures that each digit is encrypted with a different key, increasing security. Given private and public keys sk_i and pk_i respectively, for any random variable r_i, $(sk_i, pk_i) = ((k_i, p), (k_i + r_i \times p, n))$.

Encryption ($Enc(m)$) Each digit of the message is encrypted using its corresponding public key. The encryption of the message $m = m_0 m_1 \dots m_i$ is performed as: $c = m_0 \times pk_0 + m_1 \times pk_1 + \dots m_i \times pk_i$, with $pk_x < pk_y$ if $x < y$.

Decryption ($Dec(c)$) The ciphertext can be decrypted by applying the inverse process. The ciphertext is divided by the corresponding private keys, and the individual digits are recovered. This is done iteratively for each key, ensuring that each digit is processed correctly and in the proper order. The ciphertext is decrypted as: $m = \sum_{i=l}^{0}(\frac{c}{k_i}) \times 10^i, c \leftarrow c - c \times k_i$, where $(\frac{c}{k_i})$ is the quotient of $c \div k_i$.

This scheme is based on the separate processing of each decimal digit that makes up the message.

3.4.1.1 Key function

The proposed key function ensures optimal cases of keys k_i, where m is written in decimal. The keys k_i are defined as follows:

$$k_0 > 10, k_1 = 1 + 9 \times k_0, k_j = 10^{j-1} \times k_1 \tag{3.2}$$

The parameter conditions of the proposed cryptosystem are summarized as follows:

- Let: $m \in M, s = size(M)$, d is addition depth, and h is a small random number. - Keys: $k_0 > 10$, $k_j = h_j + 9 \times \sum_{i=0}^{j-1} k_i$. - For addition: $k_0 = h + 9 \times d$, $p = h' + d \times (10^{s-1} \times k_1)$.

3.4.1.2 Encryption algorithm

The encryption function encrypts the message m using the public keys pk_i.

Algorithm 3 Encryption Algorithm

Require: $m, (pk_0, pk_1, \ldots pk_i), n$
Ensure: $c = Enc(m)$

```
1: function ENC
2:     l ← size(m)
3:     c ← 0
4:     for i ← 0 to l do
5:         x ← digit(m, [l − i − 1 : l − i])
6:         c ← (c + x × (pk_i)) mod n
7:     end for
8:     return c
9: end function
```

3.4.1.3 Decryption algorithm

The decryption function decrypts the ciphertext c using the private keys k_i.

Algorithm 4 Decryption Algorithm

Require: $c, (k_0, k_1, \ldots k_j), p$
Ensure: $m = Dec(c)$

1: **function** DEC
2: $c \leftarrow c \mod p$
3: $l \leftarrow size(c)$
4: $m \leftarrow 0$
5: **for** $i \leftarrow 0$ to l **do**
6: $x \leftarrow \frac{c}{k_{l-i-1}}$
7: $c \leftarrow c - x \times k_{l-i-1}$
8: $m \leftarrow m + x \times 10^{l-i-1}$
9: **end for**
10: **return** m
11: **end function**

3.4.1.4 Homomorphic property

To ensure the additive property of the homomorphic encryption, the technique uses modular arithmetic and recursive summing of plaintext digits.

Continuing, the proposed technique from the second contribution utilizes verification proofs to ensure data integrity during homomorphically encrypted operations.

The core idea is to set the verification proof v equal to the modular arithmetic of the message.

After the client calculates the verification proof v, it encrypts the message as $c = Enc(m)$ and transmits the couple (c, v) to the cloud.

In each calculation operation O performed on c_i by the cloud, the cloud must execute the same operation O with its v_i. Finally, it computes $v_i \mod \alpha$ to obtain a single digit.

The client's verification process consists of calculating the verification proof of the result provided by the cloud. Consequently, the client decrypts c_i where $Dec(c_i) = m_i$, calculates $P(m_i)$, and checks whether v_i, the proof value returned by the cloud, is equal to $P(m_i)$.

3.4.2 Verification scheme

The core idea of this technique is to introduce a verification proof v, which is equivalent to the modular arithmetic of the message. The technique involves reducing the verification proof to a single digit for enhanced efficiency. The encryption and decryption functions, along with the verification process, are described below:

Encryption: The message m is encrypted, and the verification proof v is calculated using a small modulus α. Decryption: The ciphertext c is decrypted, and the verification proof v is verified to ensure data integrity. Homomorphic Property: The technique maintains the additive property of homomorphic encryption, allowing operations to be performed on encrypted data with integrity. Example An example illustrating the proposed cryptosystems, including an additive scheme and a multiplicative scheme, showcases the effectiveness and reliability of the proposed technique in practical scenarios. Both the client and the cloud execute operations with encrypted data while ensuring the integrity of the results through the verification process.

3.5 Performances

Cryptanalytic Insights The proposed asymmetric technique introduces two types of secret keys: the first being k_i, and the second being the trapdoor p, concealed within $n = p \times q$ (p and q being large prime numbers). If an adversary discovers p, they can extract all private keys k_i from the public keys, where $pk_i = k_i + r_i \times p$. However, obtaining the trapdoor p requires solving a factorization problem, which remains computationally infeasible. Let λ denote the security parameter, where $2^\lambda < p < 2^{\lambda+1}$. This parameter asserts that cryptanalysis necessitates 2^λ operations to find the trapdoor p.

Known-Plaintext Attack: In scenarios where the adversary possesses both the plaintext and its encrypted version (the ciphertext), known-plaintext attacks can compromise the security. However, if the known-plaintext is only one digit, the adversary gains no advantage due to the nature of the encryption process. Even with larger plaintext sizes, the structure of the encryption prevents the adversary from revealing any sensitive information easily.

Brute Force Attack (BFA): Unlike symmetric techniques where exhaustive search involves finding the private key after capturing the plaintext and ciphertext, asymmetric encryption poses a different challenge. The private keys k_i are concealed within $r \times p$, making it significantly harder to perform an exhaustive search directly on the secret keys. The complexity of finding the private key p is approximately 2^p operations.

Computational Complexity: The algorithmic complexity of the proposed technique exhibits linear behavior, with the time complexity represented as $O(M)$. This performance is compared favorably against other encryption techniques in terms of complexity, as demonstrated in Table 3.1. Moreover, the size of the ciphertext remains relatively small, contributing to enhanced efficiency.

Technique	Enc	Dec
MK-PHE	$O(\lambda)$	$O(\lambda)$
[38]	$O(\lambda^4)$	$O(\lambda^4)$
[39]	$O(\lambda^5)$	$O(\lambda^{4.8})$
[40]	$O(\lambda^6)$	$O(\lambda^5)$
[41]	$O(\lambda^6)$	$O(\lambda^5)$
[42]	$O(\lambda^{13})$	$O(\lambda^{12})$

Table 3.1: Encryption and decryption complexity comparison

Reduced Size: The symmetric version of the proposed technique offers a compact ciphertext size, as illustrated by Lemma 1. This reduction in size enhances the practical applicability of the encryption scheme.

Lemma 1. *$size(Enc(m)) = size(m) + \chi$ where χ is a constant.*

Scheme Analysis To validate the correctness and robustness of our technique, we introduce a verification condition (Lemma 2), demonstrating the preservation of integrity throughout the homomorphic encryption process. The proof leverages the homomorphic property of the system, ensuring that the computed result is accurate.

Lemma 2. *If $P(Dec(c_1 O c_2)) = v_1 O v_2$, then the returned result is correct.*

Encryption with Isolated Plaintext: While our scheme exhibits robustness against known attacks, a hypothetical scenario involving fully homomorphic encryption (FHE) reveals a potential vulnerability. In this scenario, the isolation of the target plaintext within the ciphertext poses a security risk, allowing adversaries to manipulate the result without detection. This highlights the importance of considering all possible attack vectors in designing secure encryption schemes.

Scheme Complexity: The computational complexity of our encryption scheme remains constant, irrespective of the input size. This efficiency, characterized by a single elementary calculation for both proof generation and verification, offers a significant advantage over algorithms with input size-dependent complexity.

Conclusion: The combined analysis of performance and security aspects showcases the effectiveness and reliability of the proposed encryption technique. By addressing known attacks and vulnerabilities while ensuring computational efficiency, our

scheme offers a promising solution for secure data transmission and computation in untrusted environments.

3.6 Performance, security, and scheme analysis

In this section, we delve into the performance, security, and scheme analysis of the proposed encryption technique, which combines elements of both the "A Multi-Key Based Lightweight Additive Homomorphic Encryption Scheme" and the "Scheme Analysis" research articles.

3.6.1 Security analysis

The proposed asymmetric encryption scheme introduces two types of secret keys: k_i and the trapdoor p, concealed within $n = p \times q$ where p and q are large prime numbers. The security parameter λ is crucial, ensuring cryptanalysis requires 2^λ operations to uncover the trapdoor p. Adversaries attempting to exploit known-plaintext attacks encounter formidable challenges. Even when possessing both plaintext and ciphertext pairs, the revelation of private keys remains elusive due to the complexity of factorization problems inherent in the scheme. The adversary's ability to extract private keys from public keys hinges on solving these computationally intractable problems, ensuring the scheme's robustness against known-plaintext attacks.

Moreover, brute force attacks on the proposed asymmetric encryption scheme face substantial barriers. Unlike symmetric techniques, where exhaustive searches are required to retrieve private keys, the scheme's structure imposes significant computational hurdles. With multiple secret keys k_i hidden within $r \times p$, adversaries are confronted with formidable computational complexities, greatly enhancing the scheme's resilience against brute force attacks.

3.6.2 Computational complexity

The scheme's computational complexity presents a significant advantage over existing encryption techniques. The algorithm's linear time complexity, depicted in Algorithm 1 of the original article, ensures efficient encryption and decryption processes. Notably, the time complexity $T(M) = O(M)$, where M represents the size of the message, underscores the scheme's scalability and efficiency. Comparison with other encryption techniques under similar parameters, as illustrated in

Table 3.1, highlights the superior performance of the proposed scheme, offering expedited cryptographic operations with minimal computational overhead.

3.6.3 Reduced ciphertext size

The symmetric version of the proposed encryption scheme offers a noteworthy reduction in ciphertext size, as demonstrated by Lemma 1. By leveraging a simplified formula, the ciphertext size becomes proportional to the size of the plaintext, augmented by a constant χ. This reduction in ciphertext size ensures efficient data transmission and storage while preserving the scheme's cryptographic robustness.

3.6.4 Verification condition

The scheme's integrity is fortified by a rigorous verification condition, as elucidated in Lemma 2. Leveraging the homomorphic property of the system, the verification process ensures the correctness of computation results, safeguarding against malicious manipulation by adversaries. By enforcing stringent verification conditions, the scheme maintains data integrity and authenticity, reinforcing its reliability in practical applications.

3.6.5 Application and vulnerability analysis

An exemplary application of the proposed encryption scheme in RSA underscores its efficacy in safeguarding sensitive information. By leveraging the homomorphic property, the scheme prevents unauthorized manipulation of computation results, ensuring data confidentiality and integrity. However, vulnerabilities arise in scenarios where plaintext isolation within ciphertext facilitates unauthorized manipulation. Encryption designs susceptible to such vulnerabilities underscore the importance of robust encryption schemes, resilient against adversarial exploitation.

3.6.6 Scheme complexity and efficiency

The encryption scheme's computational efficiency, characterized by $\mathcal{O}(1)$ complexity, heralds a paradigm shift in cryptographic operations. Despite input size variations, the scheme ensures consistent computational performance, offering unparalleled efficiency compared to traditional encryption algorithms. By minimizing computational overhead, the scheme optimizes resource utilization, facilitating

seamless integration into diverse applications requiring secure data transmission and processing.

In summary, the combined analysis of the proposed encryption scheme reveals its multifaceted strengths in performance, security, and efficiency. By leveraging asymmetric encryption techniques and homomorphic properties, the scheme offers robust data protection against various cryptographic attacks. Furthermore, its computational efficiency and reduced ciphertext size render it a compelling choice for secure data transmission and processing in real-world applications.

3.7 Implementation results

The implementation of the proposed encryption technique was realized using the Python programming language on a standard personal computer equipped with an Intel(R) Core(TM) i3-3110M CPU running at 2.40 GHz, featuring 2 physical cores and 4 logical processors, with 4 GB of RAM. The encryption process was evaluated using messages of 16 bits in length.

Table 3.2 provides a comparative analysis of the execution times for encryption and decryption operations of the proposed technique against four other encryption techniques.

Technique	Enc	Dec
MK-PHE	0.036	0.041
[43]	0.07	11.95
[28]	11.91	17.67
[44]	47	15
[45]	50	10
[46]	255	493
[47]	899	785

Table 3.2: Execution time comparison (milliseconds)

Experimental evaluations were conducted to assess the time efficiency of both encryption and decryption processes. With a message size of 16 bits and a modulus size of 360 bits ($n = p \times q$), our experiments revealed a notable disparity in decryption time compared to encryption time. This disparity can be attributed to two primary factors. Firstly, the decryption process involves five distinct operations (as depicted in Algorithm 4), while encryption entails only three operations (as depicted in Algorithm 3). Secondly, and perhaps more significantly, the size of the ciphertext grows at a faster rate than that of the plaintext. The decryption

process necessitates processing each digit individually, resulting in an increased computational burden influenced by the size of the ciphertext.

3.8 Conclusion

In this chapter, we have presented two distinct innovative approaches aimed at enhancing data security and integrity in cryptographic systems.

Firstly, our proposed technique introduces a novel lightweight solution with additive homomorphic encryption properties. Designed for integer numbers, its security relies on factorization and polynomial reconstruction problems. The time complexity of our technique, denoted by $T(n)$, is established as $O(n)$. We employ a straightforward encryption method for each digit m_i of the message m, defined as $c = m \times k$. The private key k_0 and p formulas determine the depth of addition. We compare the symmetric encryption of our technique with asymmetric encryption, showcasing their order-preserving properties. Security analysis of the MK-PHE scheme indicates a high level of hardness, and our implementation demonstrates superior performance in data processing speed compared to other techniques. These findings affirm that our operations are feasible even for devices with limited computing capacity, making our technique applicable in real-world scenarios.

Secondly, our introduced approach focuses on ensuring data integrity, specifically validating homomorphic calculations performed on encrypted data by untrusted third parties. Tailored for environments with constraints on time and storage space, such as the Internet of Things, our lightweight method seamlessly integrates with any fully homomorphic encryption (FHE) scheme involving encryption with non-isolated plaintext. The core innovation lies in appending a single digit to the ciphertext, thereby providing a robust mechanism for detecting tampering in results. Through foundational concepts related to the cloud and homomorphic encryption, we delineate our proposed scheme and validate its effectiveness, efficiency, and validity in preserving data integrity.

Looking ahead, we envision broad applications for these techniques across diverse fields that necessitate rigorous data security and integrity checks.

Chapter 4

Blockchain Security and Management

This chapter explores the integration of two pivotal articles in blockchain technology: "An enhanced threshold RSA-based aggregate signature scheme to reduce blockchain size, IEEE Access, 2023." and "A binary matrix-based data representation for data compression in blockchain, The 5th International Conference on Blockchain Computing and Applications (BCCA), Kuwait, Kuwait, 2023."

In the following sections, we show how these articles collectively address the issues of blockchain size and security, offering innovative solutions for optimizing storage and ensuring robust transaction verification.

4.1 Introduction

The transformative potential of blockchain technology has led to its widespread adoption across various domains, offering advantages such as enhanced data integrity, transparency, and decentralization. Blockchain has proven its efficacy in establishing trustworthy systems across diverse applications. However, as the volume of transactions recorded on blockchains continues to grow, so does the size of the blockchain itself, presenting challenges related to storage capacity and processing power.

To address this issue, the current chapter presents innovative solutions from two distinct perspectives. Firstly, we introduce a novel cryptosystem based on the RSA algorithm to provide aggregate signatures within blockchains. This scheme replaces individual transaction signatures with a single aggregate signature for each block, regardless of the number of transactions, nodes, or signers involved. By employing a flexible subgroup aggregate signature mechanism, our approach minimizes interaction between signers, thereby reducing network traffic while enhancing security robustness. Experimental analysis demonstrates the effectiveness

of our proposed aggregate signature scheme in reducing block size and overall network traffic.

Secondly, we delve into the realm of big data representation within blockchain architectures. The exponential growth of digital data generation underscores the importance of efficient data processing, transmission, and storage. To mitigate the increasing size of blockchain data, we propose a novel method for representing binary matrices within blockchains. By converting binary matrices into two vectors, we achieve significant reductions in data size while optimizing energy consumption, particularly for low-power devices. The original data can be recovered using a predefined set of vectors, offering avenues for mining consensus and data integrity verification.

In this chapter, we present pioneering approaches to address the challenges posed by blockchain size and data representation, offering insights and solutions that pave the way for advancements in blockchain technology.

4.2 Background

The prevailing design of blockchain exposes information to all network participants, making it challenging to maintain privacy and confidentiality. These issues become particularly critical in applications involving sensitive data, such as financial transactions or personal information. The concept of Secure Multiparty Computing (SMC) offers a promising solution to address this privacy concern. SMC enables multiple parties to jointly compute a function on their individual private inputs while keeping those inputs secret. Implementing SMC using blockchain, referred to as SMCB, ensures all SMC transactions are recorded as a timestamped source of truth on the blockchain.

SMCB facilitates secure data sharing and collaboration among multiple entities without requiring full trust between them. Cryptographic protocols allow participants to jointly perform computations on their private data without revealing sensitive information to other parties. This makes SMCB suitable for applications where privacy and confidentiality are paramount, such as healthcare, financial services, and supply chain management.

Encryption is a crucial aspect of information security, providing the main foundations for data confidentiality, integrity, and authentication. SMC is among the most secure encryption methods, allowing a set of distributed elements to jointly compute a random function without revealing their inputs. The development of technologies such as cloud computing, blockchain, and the Internet of Things (IoT) has increased the focus on SMC, which helps solve privacy and security issues as a general-purpose tool for calculating private data.

In the blockchain context, blocks are verified by nodes through cryptographic digital signatures. Therefore, an efficient signing process is necessary to ensure all nodes reach a consensus and verify the validity of blocks. With the widespread use of blockchain, fraudulent activities have increased, necessitating the introduction of multi-signature agreements and Aggregate Signature (AGS) schemes to reduce such incidents.

SMC properties can be exploited to create an aggregate signature that enables multiple participants to sign messages without revealing their private keys. Protocols based on multi-signature and AGS require using different keys to authorize a task, reinforcing the security of transactions more transparently. The research presents a new aggregate signature architecture based on a modified RSA cryptosystem (AGS-MR) to minimize block sizes. The model merges all signatures into a unique one, reducing the overall data size. This chapter studies its application to blockchain, demonstrating its potential to decrease block sizes significantly.

Managing large data volumes becomes a significant challenge, requiring thoughtful approaches to ensure secure and efficient data transmission, processing, and storage. Innovative technologies such as big data and cloud computing have emerged to tackle these challenges. In recent years, blockchain technology has gained considerable interest and popularity as a secure, decentralized ledger resistant to modification and censorship, making it a leading technology for storage across various applications.

However, blockchain also has its issues, particularly related to reliability and scalability. As the blockchain grows with every transaction, the storage requirements also increase, posing problems for nodes with limited capacity. This can lead to slower transaction speeds, the need for more expensive hardware, and scalability challenges.

The block size in blockchain refers to the maximum amount of data that can be stored in a single block. Large block sizes increase storage requirements and bandwidth usage, potentially leading to network centralization. Data representation significantly impacts the storage size required to store and transmit data, with different methods resulting in vastly different storage efficiencies.

The proposed method in this research utilizes binary matrices to represent blockchain data compactly. By converting transaction data into binary form and organizing it into matrices, the scheme achieves significant compression ratios. This approach not only reduces data size but also maintains the integrity and accessibility of the information.

Compression algorithms are used to reduce data size by removing redundant information. Each data representation type requires different compression methods to optimize storage and transmission efficiency. However, compression algorithms

can also increase computational overhead, affecting the blockchain network's overall efficiency. Therefore, the choice of compression algorithm depends on factors such as data characteristics, the desired level of compression, and available processing power.

One way to efficiently organize and manipulate large datasets is matrix representation, which can identify patterns and relationships in the data, thus reducing storage space. Blockchain blocks can be represented in matrices, where each row represents a transaction and each column represents a feature or variable of the transaction.

This chapter proposes a novel method of presenting and storing information in blockchain to reduce size efficiently. The approach focuses on converting a binary matrix M of size $m \times n$ into two vectors H and V of sizes m' and n', respectively. This method compresses large datasets, facilitating their transfer, use, and storage, particularly for devices with limited power. The original matrix can be reconstructed using H, V, and the matrix's hash $Hash(M)$.

The subsequent sections of this chapter are organized as follows: the next section provides a literature review of related work in the field of reduced data in blockchain. Following that, the proposed techniques and their underlying concepts are presented. An efficiency analysis is then discussed to demonstrate the feasibility of the proposed methods. Finally, a detailed discussion on the implications and potential future directions for research is provided.

4.3 Related works

Blockchain technology has undergone significant development since its inception, and numerous studies have focused on addressing its scalability, privacy, and storage challenges. These issues are particularly critical for applications generating vast amounts of data and operating on resource-constrained devices, such as those in the Internet of Things (IoT) and smart homes.

The notion of multi-signatures, introduced by Itakura and Nakamura [48], has been extensively investigated. Multi-signature schemes have been explored in various cryptographic systems, including RSA, Discrete Logarithms, Pairings, and Lattices. These studies aimed to improve multi-signature schemes concerning complexity and computational efficiency. For example, Boneh et al. [49] introduced the concept of aggregate signatures, where multiple signers can combine their signatures into a single aggregate signature, thus reducing storage overhead. However, their scheme lacked an aggregate public key, limiting its practical application.

In the blockchain context, Qiao et al. [50] proposed a short signature size aggregation signature scheme to enhance privacy and reduce storage overhead. This scheme separates the aggregate signature size from the number of signers and is based on the discrete logarithm problem, minimizing computational and verification costs. Similarly, Boneh et al. [51] developed multi-signature schemes aimed explicitly at reducing the Bitcoin blockchain size, employing techniques to prevent rogue public key attacks.

Efforts to improve the security and efficiency of blockchain include Zhao et al.'s [52] system, which combines general elliptic curve algorithms with Γ-signature schemes to create aggregate signatures. This approach optimized performance and compatibility within existing Bitcoin systems. Blanton et al. [53] combined Secure Multiparty Computation (SMC) schemes with signatures to ensure input correctness through input certification, enhancing the security of SMC applications in blockchain.

Recent advancements also include two-round multi-signature techniques and lattice-based distributed signing techniques with low round complexity. For instance, Mihir et al. [54] proposed a two-round multi-signature technique supporting key aggregation to improve security guarantees for Discrete Log-based multi-signature schemes. Other works [55, 56] explored reducing communication and computational overhead in multi-signature schemes.

Blockchain's data storage and transmission challenges have been addressed through various techniques. Summarizing and compressing blocks, storing bytecode off-chain, and using distributed storage systems like IPFS have been explored to reduce storage requirements and increase network throughput. For example, Nadiya et al. [57] applied a method to summarize and compress Bitcoin blockchain blocks, achieving significant space savings. Similarly, Norvill et al. [58] proposed a system for the Ethereum blockchain that moves the bytecode of contract creation transactions off-chain, reducing network traffic.

Segmented blockchain approaches, where nodes store only a portion of the blockchain, have also been proposed to mitigate storage demands. Xu et al. [59] introduced a segment blockchain system that uses Proof-of-Work (PoW) as a membership threshold to limit the number of nodes controlled by adversaries. However, this system faces challenges if an adversary stores all copies of a segment.

The BC-Store framework, presented by Chou et al. [60], employs an IPFS cluster system to classify hot and cold blockchain data, substantially reducing the initial synchronization time and storage requirements. Lightweight blockchain solutions, such as an improved PBFT consensus mechanism based on a reward and punishment strategy [61] and a storage optimization scheme using RS erasure code [62], have been developed to minimize computational and storage overhead further.

Privacy-preserving solutions have also been a focus of blockchain research. For instance, Yuen et al. [63] proposed a blockchain ring confidential transaction protocol (RingCT3.0) to protect the privacy of transaction participants and the confidentiality of transaction amounts.

Despite these advancements, existing solutions often have drawbacks, such as increased computational complexity, dependency on the number of signers, and potential vulnerability to certain types of attacks. Table 4.1 summarizes some of the drawbacks of previous works.

Table 4.1: A summary of the drawbacks of some previous works.

Scheme	Drawback
[49]	It did not give an aggregate public key
[51]	It uses three rounds
[64]	The verification process does not give the signers list
[55]	It needs four exponentiation operations for signing and six for verifying
[54]	Each individual signature (IS) is sent to every other signer, so there are $(n \times n)$ messages

To address these issues, this dissertation proposes a new and effective Aggregate Signature (AGS) scheme, which will be described in detail in the following sections. This scheme aims to provide a more efficient and secure method for reducing blockchain size while maintaining the integrity and security of the data.

4.4 Proposed techniques

This section outlines the proposed techniques for aggregate signature schemes and their applications in reducing data size and calculation time within a blockchain context.

4.4.1 Preliminaries

To understand the proposed techniques, we first recall some basic definitions and mathematical concepts.

- $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z} = \{0, 1, \ldots, p-1\}$: a ring of residue classes modulo p.
- pk: the public key used for encryption and verifying digital signatures.
- sk: the secret key used for encryption and decryption in symmetric systems, and for signing in asymmetric systems.
- $KeyGen$: a function that returns a pair of secret (or private) and public keys.
- mod: a function in modular arithmetic that computes the remainder after division.
- $Hashfunction$: a one-way mathematical function producing a fixed-length output from a variable-length input.
- $\times$: the multiplication operator.
- $+$: the addition operator.
- $\sum_{i=1}^{t} m_i$: denotes the sum, $m_1 + m_2 + \cdots + m_t$.
- $\prod_{i=1}^{t} m_i$: denotes the product, $m_1 \times m_2 \times \cdots \times m_t$.
- $Prime\ number$: a number $p \geq 2$ with no divisors other than 1 and itself.
- $Safe\ primes$: a prime number p such that $p = 2 \times p' + 1$ with p' also being prime.
- $Special\ RSA\ modulus$: $N = p \times q$ where p and q are safe primes.
- $Asymmetric\ scheme$: encryption using a recipient's public key, where only the corresponding private key can decrypt the message.
- $Digital\ signature$: a mechanism for message authentication using a hash function and a public-key cryptosystem.

4.4.2 AGS architecture

The AGS-MR (Aggregate Signature - Modified RSA) architecture aims to condense multiple digital signatures into a single signature, reducing the overall data size and computational overhead. Figure 4.1 illustrates the proposed modified RSA-based aggregate signature architecture.

The same RSA modulus N is used by all participants, enabling a coordinator to create an aggregate signature σ from a set of individual signatures s_i calculated using N. This approach is applicable whether the signatures are for the same message or different messages, as in blockchain transactions.

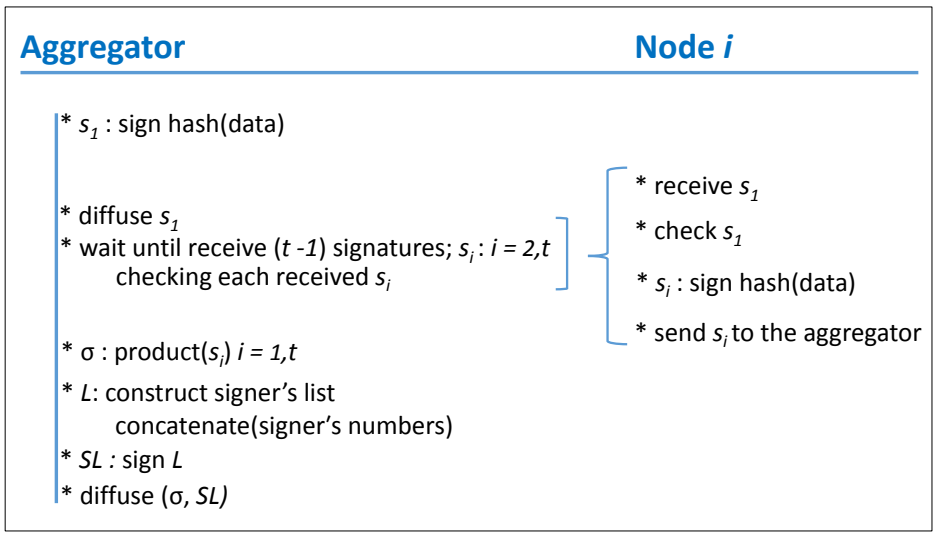

Figure 4.1: Proposed modified RSA-based aggregate signature architecture.

4.4.3 AGS-MR description

The AGS-MR scheme comprises three primary algorithms:

4.4.3.1 Key generation (KeyGen)

The Trusted Third Party (TTP) generates the public modulus N, and each node's public-private key pair (pk, sk).

```
KeyGen:
  Input: public modulus N
  Output: public-private key pair (pk, sk)
```

4.4.3.2 Signing (Sign)

Each node signs the message m using its private key sk, producing an aggregate signature σ.

```
Sign:
  Input: secret key sk, message m
  Output: aggregate signature
```

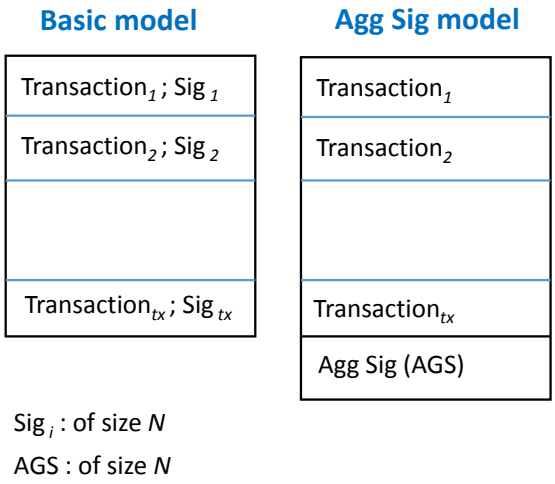

Figure 4.2: Basic and aggregate signature block models.

4.4.3.3 Verification (Verify)

The verifier checks the validity of the signature σ using the public key pk, the message m, and the signature σ.

```
Verify:
  Input: public key pk, message m, signature
  Output: bit 1 (accepted) or 0 (rejected)
```

4.4.4 Process description

The proposed AGS-MR process involves several steps, as depicted in Figure 4.2.

4.4.4.1 Setup

The TTP sets up the public parameters, including the RSA modulus N.

4.4.4.2 Key generation

The TTP generates the node's public and private keys (e, d), keeping p and q secret to prevent nodes from generating their own private keys.

4.4.4.3 Signing: First round

The aggregator node signs the data and diffuses the signature s_1 on the network. Other nodes verify s_1, sign the message, and send their signatures back to the aggregator.

Algorithm 5 Individual Signature Algorithm

Require: $(message : M;\ sig_{aggregator} : s_1;\ pk_{aggregator} : e_1);\ sk : d'_i;\ threshold : t;\ pub\ param : e_r, N$
Ensure: $sig : s_i$

function NODE$_i$ SIG
  $h \leftarrow hash(M)$
  $y \leftarrow h^{e_r \times t + e_1} \mod N$
  $y' \leftarrow s_1^{e_r \times t \times e_1} \mod N$
  if $y = y'$ **then** ▷ verify aggregator signature validity
    $s_i \leftarrow h^{d'_i} \mod N$ ▷ compute its own signature
    send s_i to the aggregator
  else
    invalid aggregator signature
  end if
end function

4.4.4.4 Signing: Second round

The aggregator collects all signatures and computes the aggregate signature σ by multiplying the individual signatures. The list of participants is also signed by the aggregator and broadcasted on the network.

$$\sigma = \prod_{i=1}^{t} s_i \tag{4.1}$$

4.4.4.5 Aggregate signature verification

Anyone can verify the aggregate signature using the public parameters, the message, and the list of participants.

The correctness of the aggregate signature verification process ensures that if all individual signatures are valid, the aggregate signature will also be valid.

Algorithm 6 AGS Aggregator Algorithm

Require: $message : M;\ pk(s) : e_i;\ sk : d'_1;\ threshold : t;\ pub\ param : e_r, N$
Ensure: $sig : \sigma,\ signer's\ list : SL$

function AGGREGATOR SIG
  $h \leftarrow hash(M)$
  $s_1 \leftarrow h^{d'_1} \mod N$
  diffuse (M, s_1)
  $list \leftarrow \{\}$
  $L \leftarrow''$
  while $|list| < t$ **do**
    receive sig s_i of node n_i
    checking the validity of each s_i
    insert n_i in $list$
    $L \leftarrow L \parallel n_i$
  end while
  $\sigma \leftarrow \prod_{i=1}^{t} s_i \mod N$
  $SL \leftarrow (hash(L))^{d'_1} \mod N$
  diffuse (M, σ, L, SL)
end function

4.4.5 Efficient data representation technique

In this section, we present two innovative data representation techniques aimed at improving efficiency and security in data storage and retrieval.

The first technique focuses on efficiently representing data to reduce storage requirements and facilitate quicker access and processing. This method is particularly useful in scenarios where large datasets need to be stored and accessed frequently, such as in cloud storage or big data analytics.

4.4.5.1 Key features

- **Matrix-Based Compression**: Data is organized into a matrix format, which is then compressed using a novel algorithm that identifies and eliminates redundancy.
- **Vector Extraction**: The technique extracts two vectors from the matrix—one representing the number of "1"s in each row and the other representing the number of "1"s in each column. This reduces the size of the data representation.

Algorithm 7 AGS Verification Algorithm

Require: $message : M$; $pk_{aggregator} : e_1$; $agg\ sig : \sigma$; $signerslist : L$; $signed\ list : SL$; $pub\ param : t, e_r, N$
Ensure: $sig : s_i$

```
function VERF SIG
    h ← hash(M)
    Check the validity of the signer's list
    Extract participating signers
    h' ← h^{e_r}
    Calculate s using public keys
    Calculate s' using σ and public keys
    if s = s' then
        Accept the aggregate signature
    else
        Reject the aggregate signature
    end if
end function
```

- **Hashing for Uniqueness**: To ensure the uniqueness and integrity of the data, a hash of the original matrix is included in the representation.

4.4.5.2 Steps involved

1. **Matrix Conversion**: Convert the original data into a matrix format.
2. **Vector Generation**: Generate horizontal and vertical vectors representing the number of "1"s in each row and column, respectively.
3. **Hash Calculation**: Compute the hash of the original matrix to maintain data integrity and uniqueness.
4. **Data Storage**: Store the vectors and the hash instead of the entire matrix, significantly reducing the storage size.

4.4.6 The proposed data representation concept

In this part, we present a detailed explanation of the proposed technique, which aims to present the data in a very compact way. The node converts the matrix containing the data according to the proposed method and then sends it. The conversion process is very easy and fast regardless of the size of the matrix; also the converted output is small in size compared to the original data size.

4.4.6.1 Representation of the proposed model

In the proposed model, we define a new method of presenting and storing information to reduce the size very efficiently. This method extracts two vectors, horizontal H and vertical V, which replace the matrix. Suppose we have a matrix M containing m rows and n columns. The first vector H is a vector containing m values, the first value is the number of "1"s in the first row of the matrix, the second value is the number of "1"s in the second row, and so on until the last value, that is the number of "1"s in the last row of the matrix.

Similarly, for the second vector V, it is a vector that contains n values, the first value is the number of "1"s in the first column of the matrix, the second value is the number of "1"s in the second column, and so on until the last value, that is the number of "1"s in the last column of the matrix.

Representing the information in this way is insufficient to retrieve it from the cloud because we can find two different matrices A and B with the same vector representation, i.e., $H_A = H_B$ and $V_A = V_B$. Therefore, in the proposed model, we add the hash of the matrix to represent data uniquely, where $Hash_A \neq Hash_B$. The following example illustrates this concept in a simple way.

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

with $H_A = (2, 2, 2, 2)$; $V_A = (2, 1, 2, 3)$.

and

$$B = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

with $H_B = H_A = (2, 2, 2, 2)$; $V_B = V_A = (2, 1, 2, 3)$.

4.4.6.2 Proposed model: Recuperation

After representing a matrix M as mentioned in Subsection 4.4.6.1, the cloud can use its available computing power to recover the original matrix based on H_M, V_M, and $Hash_M$. The currently used algorithm in the proposed model is based on

searching exhaustively according to the possibilities that exist. When the search starts from a zeros matrix with m rows and n columns, then the existing possibilities are tested based on the number of "1"s in the two vectors H_M and V_M. For each matrix possibility M_i calculated, the cloud computes its hash and compares it with $Hash_M$. Algorithm 8 gives a general illustration of this procedure.

Algorithm 8 Recover matrix algorithm

Require: $H_M, V_M, Hash_M$
Ensure: $matrixM$

```
1: function RecM
2:     M_0 ← zeros
3:     for for each possible composition of the number of 1 do
4:         form M_i
5:         compute H_i
6:         compute V_i
7:         compute Hash_i
8:         if H_i = H_M and V_i = V_M and Hash_i = Hash_M then
9:             M ← M_i
10:            exit
11:        end if
12:    end for
13:    return M
14: end function
```

Now, we come to the number of possible matrices (NPM) that the algorithm 8 will extract from H and V.

Lemma 3. *If* $size(M) = n \times m$ *then* $NPM_{n,m} = \prod_{i=1}^{m} NPR_i$ *where* m *is the number of rows and* NPR_i *denotes the number of possibilities in the row* i*.*

We give $NPR_i = \alpha \times (\alpha + 1) \times (2 \times \alpha + 4)/12$ *where* $\alpha = n - x + 1$

Proof. The *RecM* algorithm 8 will not try all possibilities from 'zeros' to 'ones'; in this case, the number of possibilities is $2^{n \times m}$. This is because the number of possible values for the first line is 2^n if the first line size is equal to n, and if the matrix has m rows, then $NPM = 2^n \times 2^n \ldots 2^n = 2^{n \times m}$.

The *RecM* algorithm 8 will exploit the number of 'ones' in each line to reduce the total number of possibilities, thus greatly reducing the number of trials.

Suppose the number of "1" is x with $x < n$ where n is the size of a row; the first possibility is to position the x bits of "1" successively (e.g. 11100000000 where x

= 3 and $n = 11$). The next possibility is to move the last "1" (11010000000; then 11001000000 ..) etc.

Therefore, we have $(n - x + 1)$ possibilities. Next, we move the "1" before the last one and do the same with the last "1" as the first step (10110000000; 10101000000; 10100100000), so there are $(n - x + 1) - 1$ possibilities.

In the third step, we do the same thing with the first "1", we will finally find that NPR is equal to the sum of the series $1+(1+2)+(1+2+3)\cdots+(1+2+3+4+\cdots+\alpha)$ where α is $(n - x + 1)$ and x denotes the number of "1" in the first row.

We have $s_i = \sum_{i=1}^{i} i = i \times (i+1)/2$ and $s_{i+1} = \sum_{i=1}^{i+1} s_i = i \times (i+1) \times (2i+4)/12$; then the number of possible rows is $NPR_i = s_{i+1} = \alpha \times (\alpha+1) \times (2 \times \alpha + 4)/12$.

The number of possible matrices is the product of the m rows possibilities. □

In summary, this section presented the AGS-MR scheme, an efficient aggregate signature method based on a modified RSA architecture. The proposed scheme reduces data size and computation time, making it suitable for blockchain applications.

4.5 Performance and security analysis

4.5.1 Performance experimental analysis

To evaluate the performance of the proposed AGS-MR, this section is divided into three main subsections: an analysis of execution time, scalability, and data size.

4.5.1.1 Execution time

The proposed AGS-MR was implemented in Python running on a personal computer with an Intel(R) Core(TM) i3-3110M CPU 2.40 GHz, 2 Core(s), 4 Logical Processor(s), and 4 GB RAM. Each node was simulated by a process where multiprocess programming was implemented.

Table 4.2 and Figure 4.3 display the results of signing and verification function executions. In these experiments, the time to build and extract the signers' list in the Sig and Verf phases was ignored. The implementation settings were $N = 32$ bits, t is the required number of signers, and SHA256 hash was used. In each test, a secret key d_r was chosen, factorized, and t and r were extracted, where $d_r = t \times r$.

Table 4.2: AGS-MR execution time with different numbers of required signatures.

# signatures	Sig (ms)	Verf (ms)
13	0.06	0.22
29	0.13	0.58
117	0.54	7.39
203	0.93	23.4
351	1.62	74.3
1053	5.04	925

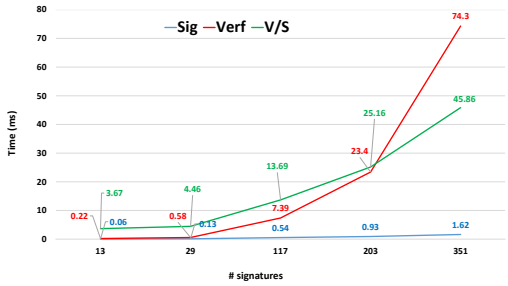

Figure 4.3: Execution time of signing and verifying processes for different numbers of signatures.

4.5.1.2 Scalability

Scalability refers to the system's ability to accommodate the number of signers. The proposed AGS-MR demonstrates scalability, as the size of the AGS remains constant regardless of the number of signees. The list of signers in the threshold system is discussed, where t signers are needed from n elements.

4.5.1.3 Network traffic

Cooperative blockchain techniques, such as multi-signature schemes, can be categorized into interactive and non-interactive schemes. Interactive schemes involve complete interaction among all participants, whereas non-interactive schemes minimize system traffic by requiring only partial interaction. The proposed AGS-MR falls into the non-interactive category, generating only $t + n$ messages.

4.5.1.4 Data size

The proposed AGS-MR introduces a compact aggregate signature technique with public key aggregation. The technique allows any subset s of a set S to sign a message m, revealing which subset generated the signature. The signature size is only $O(k)$ bits independent of t and x, where $t = |s|$ and $x = |S|$.

4.5.2 Security analysis

This section discusses the security model, the rationale behind using modified RSA, and methods to prevent attacks on blockchain.

4.5.2.1 Security model

The proposed technique employs a trusted third party (TTP) to select large safe prime numbers p and q, keeping them secret. Each node i is provided with a modified RSA key pair (e_i, d_i'), where $d_i' = d_i + r$. The security relies on the difficulty of factoring $N = p \times q$ and the collision resistance of the chosen hash function.

4.5.2.2 Why modified RSA?

The modification of RSA keys prevents rogue public key attacks and enhances security. Each key pair (e_i, d_i') is authenticated by the TTP, and the addition of r ensures that d_i' cannot be directly derived from e_i.

4.5.2.3 Preventing attacks on blockchain

The proposed technique mitigates 51% and rogue public key attacks by requiring t' signatures to validate a block, making it computationally intensive for malicious actors to manipulate the system.

4.6 Security analysis and efficiency

In this section, we analyze the security properties and efficiency of the proposed AGS-MR scheme.

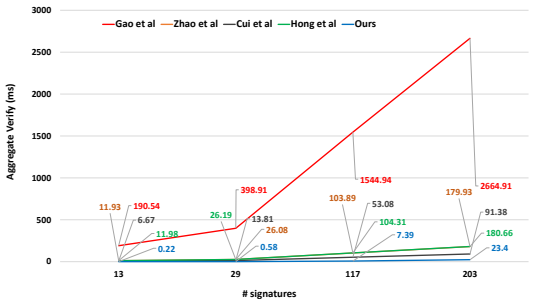

Figure 4.4: Comparison of individual and aggregate signature verification times.

4.6.1 Security analysis

The security of the AGS-MR scheme relies on the difficulty of factoring large RSA moduli and the hardness of the hash function. The following security properties are considered:

- **Unforgeability**: It is computationally infeasible for an adversary to forge a valid aggregate signature without knowing the private keys of the participating nodes.
- **Non-repudiation**: A signer cannot deny the validity of their signature since it is uniquely linked to their private key.
- **Integrity**: Any modification to the message or the aggregate signature will result in a verification failure.

4.6.2 Efficiency

The efficiency of the AGS-MR scheme is evaluated in terms of data size and computational overhead. By aggregating individual signatures, the overall data size is significantly reduced. Additionally, the verification process is streamlined, requiring fewer computations compared to verifying each signature individually.

4.7 Conclusion

In this chapter, we have explored two pioneering advancements in the realms of cryptographic protocols and blockchain technology.

Firstly, we introduced AGS-MR, an innovative aggregate signature scheme derived from the RSA cryptosystem. By enhancing the fundamental RSA technique, AGS-MR ensures heightened security for scenarios requiring multiple signatures without nodes needing to prove knowledge or possess their secret key. Notably, AGS-MR achieves a final aggregate signature size of $O(k)$, irrespective of network size, thereby significantly reducing communication and computation costs compared to existing schemes. Our comparative analysis showcases AGS-MR's superiority over competitors, such as Boneh AMSP and Cui et al. Future research avenues will explore further optimizations for the computation of the aggregate public key, aiming to ensure scalability and efficiency across various signer configurations.

Secondly, we proposed a groundbreaking method for compressing big data within blockchain architectures. Our approach addresses the challenges posed by the substantial data size inherent in blockchain technology, offering a simple yet effective compression model to facilitate seamless data transfer, utilization, and storage. Empirical evaluations demonstrate remarkable reductions in data size, with effectiveness scaling with the original data's magnitude. Future work will leverage our compression method to devise a consensus system focused on original data recovery and advocate for the establishment of a central system for storing and verifying the integrity of original data.

In summation, our exploration of these two advancements underscores the potential for transformative progress in cryptographic protocols and blockchain technology, presenting a multitude of opportunities and solutions for future research and implementation.

Conclusion

In the dynamic landscape of modern computing, where the fusion of IoT devices, cloud computing, and blockchain technology is reshaping the fabric of our digital ecosystem, the imperative for robust data security and privacy has never been more pronounced. The research presented in this dissertation underscores the multifaceted challenges of securing distributed environments and introduces innovative solutions to enhance their efficiency, reliability, and resilience.

From the introduction of lightweight homomorphic encryption schemes tailored for resource-constrained IoT devices to the development of efficient integrity verification mechanisms for cloud-executed operations, the research efforts outlined in this work have focused on strengthening the security and management of distributed systems. These contributions do not directly apply data analysis or machine learning techniques but instead provide the necessary foundations to ensure that future applications of efficient data analysis and machine learning in IoT environments can be executed securely and effectively.

Additionally, the exploration of novel data compression techniques for blockchain networks and the integration of cryptographic primitives into IoT ecosystems signal a shift toward decentralized, scalable, and secure data management. By addressing fundamental security and efficiency concerns, these contributions lay the groundwork for the seamless integration of advanced analytical and computational models in real-world applications.

In conclusion, this dissertation serves as a pivotal step toward fortifying the security of distributed systems, thereby enabling the broader adoption of data-driven technologies in a secure and efficient manner. By leveraging cutting-edge cryptographic techniques and innovative approaches to data security and privacy, this research provides the structural backbone necessary for future advancements in machine learning and data analysis within IoT ecosystems. As the digital landscape continues to evolve, the insights and methodologies presented here will help shape a more secure, trustworthy, and resilient technological future.

Outlook

The research presented in this dissertation lays a solid foundation for future exploration and innovation in the realm of data security, privacy, and integrity. Building upon the insights gained from the four key contributions, several promising avenues for further research and development emerge:

1. **Scalable cryptographic solutions**: As the volume and complexity of digital data continue to grow exponentially, there is a pressing need for scalable cryptographic solutions that can effectively address the evolving threat landscape. Future research efforts could focus on the development of novel encryption techniques, signature schemes, and authentication protocols that strike a balance between security, efficiency, and scalability.

2. **Interdisciplinary collaboration**: The intersection of cryptography, distributed systems, and emerging technologies such as blockchain and IoT presents rich opportunities for interdisciplinary collaboration. Future research endeavors could leverage insights from diverse fields, including computer science, mathematics, economics, and law, to develop holistic solutions to complex security and privacy challenges.

3. **Privacy-preserving technologies**: With increasing concerns about data privacy and surveillance, there is a growing demand for privacy-preserving technologies that empower individuals to retain control over their personal information. Future research could explore advanced techniques such as zero-knowledge proofs, differential privacy, and secure multiparty computation to enable secure and privacy-preserving data sharing and analysis.

4. **Real-world deployment and evaluation**: While theoretical advancements in cryptography are essential, their real-world impact ultimately depends on their practical deployment and evaluation. Future research efforts should prioritize the implementation and deployment of cryptographic solutions in diverse real-world settings, allowing for empirical evaluation of their performance, scalability, and security.

5. **Ethical and regulatory considerations**: As cryptographic technologies become increasingly integrated into our daily lives, it is imperative to consider the ethical, legal, and regulatory implications of their deployment. Future research could explore frameworks for ethical decision-making in cryptography, as well as mechanisms for ensuring compliance with privacy regulations and standards.

By embracing these outlooks and continuing to push the boundaries of innovation and collaboration, researchers and practitioners can collectively work towards

building a more secure, privacy-preserving, and resilient digital infrastructure for the benefit of society as a whole.

Bibliography

[1] **Khaled Chait**, Abdelkader Laouid, Mostefa Kara, Mohammad Hammoudeh, Omar Aldabbas, and Abdullah T. Al-Essa. An enhanced threshold rsa-based aggregate signature scheme to reduce blockchain size. *IEEE Access*, 11:110490–110501, 2023. https://doi.org/10.1109/ACCESS.2023.3322196.

[2] Brahim Ferik, Lakhdar Laimeche, Abdallah Meraoumia, Abdelkader Laouid, **Khaled Chait**, and Muath AlShaikh. A blind digital watermarking approach using palmprint-embedded local binary patterns and arnold cat map transformations. In *2024 6th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pages 1–8. IEEE, 2024. https://doi.org/10.1109/PAIS62114.2024.10541138.

[3] Berhoum Adel, Mohammad Charaf Eddine Meftah, Abdelkader Laouid, **Khaled Chait**, and Mostefa Kara. Using transformers to classify arabic dialects on social networks. In *2024 6th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pages 1–7. IEEE, 2024. https://doi.org/10.1109/PAIS62114.2024.10541289.

[4] **Khaled Chait**, Mostefa Kara, Abdelkader Laouid, Mohammad Hammoudeh, and Ahcène Bounceur. One digit checksum for data integrity verification of cloud-executed homomorphic encryption operations. In *Proceedings of the 7th International Conference on Future Networks and Distributed Systems*, ICFNDS '23, pages 71–75. ACM, 2023. https://doi.org/10.1145/3644713.3644724.

[5] Abdelkader Laouid, Mostefa Kara, Mohammed Al-Khalidi, **Khaled Chait**, Mohammad Hammoudeh, and Ahmed Aziz. A binary matrix-based data representation for data compression in blockchain. In *The 5th International Conference on Blockchain Computing and Applications (BCCA2023)*, pages 307–314. IEEE, 2023. https://doi.org/10.1109/BCCA58897.2023.10338911.

[6] Sana Sahar Guia, Abdelkader Laouid, and **Khaled Chait**. Categorization of digital pathology image using deep learning model. In *Proceedings of the 7th International Conference on Future Networks and Distributed Systems*, ICFNDS '23, pages 397–402. ACM, 2023. https://doi.org/10.1145/3644713.3644769.

[7] Brahim Ferik, Lakhdar Laimeche, Abdallah Meraoumia, Abdelkader Laouid, Muath AlShaikh, and **Khaled Chait**. An adaptive image watermarking based on bellman-ford algorithm. In *Proceedings of the 7th International Conference on Future Networks and Distributed Systems*, ICFNDS '23, pages 357–365. ACM, 2023. https://doi.org/10.1145/3644713.3644762.

[8] **Khaled Chait**, Abdelkader Laouid, Lamri Laouamer, and Mostefa Kara. A multi-key based lightweight additive homomorphic encryption scheme. In *2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP)*, pages 1–6. IEEE, 2021. https://doi.org/10.1109/AI-CSP52968.2021.9671216.

[9] Jorge Guajardo and Christof Paar. Efficient algorithms for elliptic curve cryptosystems. In *Advances in Cryptology — CRYPTO '97*, pages 342–356, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[10] Hamad Sarah Shihab and Sagheer Ali Makki. Design of fully homomorphic encryption by prime modular operation. *Telfor Journal*, 10(2):118–122, 2018.

[11] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[12] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[13] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[14] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, EUROCRYPT'99, pages 223–238. Springer, Springer-Verlag, 1999.

[15] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of Cryptography: Second Theory of Cryptography Conference, TCC 2005*, pages 325–341, Berlin, Heidelberg, feb 2005. Springer, Springer Berlin Heidelberg.

[16] Josh D. Cohen and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, SFCS '85, page 372–382, USA, 1985. IEEE Computer Society.

[17] Ben Adida. Helios: Web-based Open-Audit voting. In *17th USENIX Security Symposium (USENIX Security 08)*, San Jose, CA, jul 2008. USENIX Association.

[18] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, page 187–196, New York, NY, USA, 2008. Association for Computing Machinery.

[19] E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, page 364. IEEE Computer Society, 1997.

[20] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978.

[21] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178. Association for Computing Machinery, 2009.

[22] Chris Peikert. A decade of lattice cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4):283–424, mar 2016.

[23] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, oct 1997.

[24] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), sep 2009.

[25] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, Stanford, CA, USA, 2009.

[26] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques*, pages 24–43, Berlin, Heidelberg, 2010. Springer, Springer Berlin Heidelberg.

[27] Mostefa Kara, Abdelkader Laouid, Muath AlShaikh, Mohammad Hammoudeh, Ahcene Bounceur, Reinhardt Euler, Abdelfattah Amamra, and Brahim Laouid. A compute and wait in pow (cw-pow) consensus algorithm for preserving energy consumption. *Applied Sciences*, 11(15):6750, 2021.

[28] Hongping Pang and Baocang Wang. Privacy-preserving association rule mining using homomorphic encryption in a multikey environment. *IEEE Systems Journal*, 2020.

[29] Yarkın Doröz, Aria Shahverdi, Thomas Eisenbarth, and Berk Sunar. Toward practical homomorphic evaluation of block ciphers using prince. In *Financial Cryptography and Data Security*, volume 8438, pages 208–220, Berlin, Heidelberg, 03 2014. Springer Berlin Heidelberg.

[30] Alexander Viand, Christian Knabenhans, and Anwar Hithnawi. Verifiable fully homomorphic encryption. *arXiv preprint arXiv:2301.07041*, 2023.

[31] Ruwei Huang, Zhikun Li, and Jianan Zhao. A verifiable fully homomorphic encryption scheme. In *Security, Privacy, and Anonymity in Computation, Communication, and Storage*, pages 412–426. Springer International Publishing, 2019.

[32] Ahmed El-Yahyaoui and Mohamed Kettani. A verifiable fully homomorphic encryption scheme for cloud computing security. *Technologies*, 7(1):21, 2019.

[33] Dario Fiore, Rosario Gennaro, and Valerio Pastro. Efficiently verifiable computation on encrypted data. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 844–855. Association for Computing Machinery, 2014.

[34] Fangyuan Jin, Yanqin Zhu, and Xizhao Luo. Verifiable fully homomorphic encryption scheme. In *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pages 743–746. IEEE, 2012.

[35] Fucai Luo and Kunpeng Wang. Verifiable decryption for fully homomorphic encryption. In *Information Security: 21st International Conference, ISC 2018, Guildford, UK, September 9–12, 2018, Proceedings 21*, pages 347–365. Springer, 2018.

[36] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 309–325, New York, NY, USA, 2012. Association for Computing Machinery.

[37] Ruba Awadallah, Azman Samsudin, and Mishal Almazrooie. Verifiable homomorphic encrypted computations for cloud computing. *International Journal of Advanced Computer Science and Applications*, 12(10), 2021.

[38] Keke Gai, Meikang Qiu, Yujun Li, and Xiao-Yang Liu. Advanced fully homomorphic encryption scheme over real numbers. In *2017 IEEE 4th international*

conference on cyber security and cloud computing (CSCloud), pages 64–69. IEEE, 2017.

[39] V Biksham and D Vasumathi. A lightweight fully homomorphic encryption scheme for cloud security. *International Journal of Information and Computer Security*, 13(3-4):357–371, 2020.

[40] Hao-Miao Yang, Qi Xia, Xiao-fen Wang, and Dian-hua Tang. A new somewhat homomorphic encryption scheme over integers. In *2012 International Conference on Computer Distributed Control and Intelligent Environmental Monitoring*, pages 61–64. IEEE, 2012.

[41] Y Govinda Ramaiah and G Vijaya Kumari. Towards practical homomorphic encryption with efficient public key generation. *International Journal on Network Security*, 3(4):10, 2012.

[42] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrede Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 315–335. Springer, 2013.

[43] Mostefa Kara, Abdelkader Laouid, Reinhardt Euler, Mohammed Amine Yagoub, Ahcene Bounceur, Mohammad Hammoudeh, and Saci Medileh. A homomorphic digit fragmentation encryption scheme based on the polynomial reconstruction problem. In *The 4th International Conference on Future Networks and Distributed Systems (ICFNDS)*, pages 1–6, 2020.

[44] M Thangavel and P Varalakshmi. Enhanced dna and elgamal cryptosystem for secure data storage and retrieval in cloud. *Cluster Computing*, 21(2):1411–1437, 2018.

[45] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, pages 487–504, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[46] Smaranika Dasgupta and S.K. Pal. Design of a polynomial ring based symmetric homomorphic encryption scheme. *Perspectives in Science*, 8, 07 2016.

[47] Derian Boer and Stefan Kramer. Secure sum outperforms homomorphic encryption in (current) collaborative deep learning. *CoRR*, abs/2006.02894, 06 2020.

[48] Kazuharu Itakura and Katsuhiro Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, 1(71):1–8, 1983.

[49] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiability encrypted signatures from bilinear maps. In *Advances in Cryptology — EUROCRYPT 2003*, pages 416–432, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[50] Kang Qiao, Hongbo Tang, Wei You, and Yu Zhao. Blockchain privacy protection scheme based on aggregate signature. In *2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pages 492–497. IEEE, 2019.

[51] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 435–464. Springer, 2018.

[52] Yunlei Zhao. Practical aggregate signature from general elliptic curves, and applications to blockchain. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 529–538, 2019.

[53] Marina Blanton and Myoungin Jeong. Improved signature schemes for secure multi-party computation with certified inputs. In *European Symposium on Research in Computer Security*, pages 438–460. Springer, 2018.

[54] Mihir Bellare and Wei Dai. Chain reductions for multi-signatures and the hbms scheme. In *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part IV*, pages 650–678. Springer, 2021.

[55] Kwangsu Lee and Hyoseung Kim. Two-round multi-signatures from okamoto signatures. *Mathematics*, 11(14), july 2023.

[56] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. *Journal of Cryptology*, 35(2):1–56, 2022.

[57] Ulfah Nadiya, Kusprasapta Mutijarsa, and Cahyo Y Rizqi. Block summarization and compression in bitcoin blockchain. In *2018 International Symposium on Electronics and Smart Devices (ISESD)*, pages 1–4. IEEE, 2018.

[58] Robert Norvill, Beltran Borja Fiz Pontiveros, Radu State, and Andrea Cullen. Ipfs for reduction of chain size in ethereum. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1121–1128. IEEE, 2018.

[59] Yibin Xu and Yangyu Huang. Segment blockchain: A size reduced storage mechanism for blockchain. *IEEE Access*, 8:17434–17441, 2020.

[60] I-Te Chou, Hung-Han Su, Yu-Ling Hsueh, and Chih-Wen Hsueh. Bc-store: A scalable design for blockchain storage. In *Proceedings of the 2nd International Electronics Communication Conference*, pages 33–38, 2020.

[61] Chunlin Li, Jing Zhang, Xianmin Yang, and Luo Youlong. Lightweight blockchain consensus mechanism and storage optimization for resource-constrained iot devices. *Information Processing & Management*, 58(4):102602, 2021.

[62] Hong Shu, Ping Qi, Yongqing Huang, Fulong Chen, Dong Xie, and Liping Sun. An efficient certificateless aggregate signature scheme for blockchain-based medical cyber physical systems. *Sensors*, 20(5):1521, 2020.

[63] Tsz Hon Yuen, Shi-feng Sun, Joseph K Liu, Man Ho Au, Muhammed F Esgin, Qingzhao Zhang, and Dawu Gu. Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*, pages 464–483. Springer, 2020.

[64] Rakyong Choi and Kwangjo Kim. Lattice-based multi-signature with linear homomorphism. In *2016 Symposium on Cryptography and Information Security (SCIS 2016)*. The Institute of Electronics, Information and Communication Engineers (IEICE), 2016.