Achieving Communication-efficient Privacy-preserving Range Query in Fog-based IoT

by

Hassan Mahdikhani

Master of Computer (Software Engineering), Iran University of Science and Technology, Iran, 2006
Bachelor of Computer (Software Engineering), Kharazmi University, Iran, 2001

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

In the Graduate Academic Unit of Computer Science

Supervisor(s):	Rongxing Lu,Ph.D., Faculty of Computer Science, Ali Ghorbani, Ph.D., Faculty of Computer Science
Examining Board:	Suprio Ray, Ph.D., Faculty of Computer Science, Kalikinkar Mandal, Ph.D., Faculty of Computer Science, Zhen Lei, Ph.D., Faculty of Civil Engineering
External Examiner:	Rei Safavi-Naini, Ph.D., Faculty of Computer Science, University of Calgary

This dissertation is accepted by the
Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

February, 2021

Abstract

Fog-based IoT (Internet of Things) is a fast-growing technology in which many firms and industries are currently investing to develop their own real-time and low latency decentralized data processing and analysis applications. It narrows down the gap between cloud and IoT end-devices as cloud computing is not a consistently perfect solution for many IoT applications. Compared with the traditional IoT solutions, fog-enabled IoT can offer a high level of compliance, better efficiency, and stronger security by providing local data pre-processing, filtering, and forwarding mechanisms. These benefits make the fog-enhanced IoT an appropriate paradigm for many IoT services in different applications varying from health monitoring systems to smart grids and even food manufacturing. However, fog-enhanced IoT arises many security and privacy concerns since fog nodes are deployed at the network edge and may not be fully trustable. Furthermore, fog is considered as a non-trivial extension of the cloud, and thus some security and privacy challenges will continue to persist. These challenges might affect the adaptation of fog computing into the IoT. At the same time, fog improves the IoT end-devices' security and privacy by offering an ideal platform to employ homomorphic encryption schemes. Homomorphic encryption schemes allow performing mathematical operations on ciphertexts without violating the IoT devices' privacy. This means that instead of separately delivering each IoT device's data to the control center, the fog nodes can forward the encrypted aggregated results. This alternative approach significantly reduces the communication

overhead and greatly strengthens the security robustness. Thus, system developers can design data aggregation algorithms that yield more bandwidth-efficient, secure, and private schemes than traditional cloud deployment. In this thesis, we emphasize on range aggregate queries in fog-enhanced IoT. In particular, we carry on research on communication and computational efficient privacy-preserving range query processing schemes in which the querying user can efficiently execute range queries on IoT end-devices in the fog computing environment. The main contributions of this thesis can be summarized as 1) Taking the computational burden into consideration, we devise an efficient Symmetric Homomorphic Encryption (SHE) scheme. The proposed scheme maintains data privacy and security as well as supports homomorphic calculation in arithmetic circuits including both multiplication and addition operations. 2) To achieve higher communication performance, we develop some range decomposition/composition techniques to transfomr the range queries. These techniques transform a given range query $[L, U]$ into corresponding data structures that realize privacy-preserving communication-efficient range aggregate query protocols. We develop three different decomposition/composition schemes and investigate their computational and communication performance. 3) Analysing the security of these developed schemes to ensure that proposed schemes are privacy-preserving, i.e. querying user's query and IoT end-devices' data can not be identified or profiled by not only fraudulent/dishonest but also honest-but-curious entities. 4) Conducting extensive performance evaluations to demonstrate the effectiveness of the proposed schemes in terms of communication outcomes and computational effort reduction.

Dedication

To my family and teachers without whom, I would have achieved nothing.

Acknowledgements

My first appreciation goes to my supervisor Dr. Rongxing Lu, for his continuous encouragement and precious guidance to facilitate a good quality of research and publications. He spent a significant amount of time on my regular supervision sessions throughout my Ph.D., he especially maintained a very pleasant creative atmosphere to effectively interact with him and other scholars.

Also, many thanks to my co-supervisor, Dr. Ali Ghorbani, for his time and outstanding leadership towards the completion of my research and Ph.D.

My gratitude goes to my thesis advisory committee members, Dr. Suprio Ray, Dr. Kalikinkar Mandal, Dr. Zhen Lei, and Dr. Rei Safavi-Naini, for patiently reading through my thesis, and providing valuable feedback that has served to improve this thesis.

I would like to express my sincere thanks to Dr. Jun Shao for his support and for providing valuable discussions and feedback to publish papers that create the basis for this thesis and publishing two journal papers. I am also grateful to Dr. Suprio Ray, Dr. David Bremner, Dr. Rasoul Shahsavari, Mr. Shlomi Linoy, Dr. Hui Zhu, Dr. Arash Habibi Lashkari, Evan R. Kennedy, and Dr. Natalia Stekhonva for their valuable support and their contributions to the co-authored papers and research studies.

Moreover, for suggestions, discussions, and presentations I thank my friends, graduate and Ph.D. students in our research group under Dr. Lu's supervision, with

special gratitude to Dr. Xiong Li, Dr. Beibei Li, Dr. Xue Yang, Mr. Wei Guo, Ms. Yandong Zheng, Mr. Yunguo Guan, Mr. Xicheng Zhang, Mr. Songnian Zhang, Ms. Jiacheng Jin, Ms. Dan Zhu, and Ms. Yaxi Yang. I am also thankful to all my fellow from the Canadian Institute for Cybersecurity at the University of New Brunswick for their support.

Table of Contents

List of Tables

List of Figures

Abbreviations

BCP Bresson, Catalano, and Pointcheval cryptosystem. 23

BF Bloom Filter. viii, 33

BGN Boneh, Goh, and Nissim cryptosystem. 8

CCA Choosen Ciphertext Attack. 58

COA Ciphertext Only Attack. 58

CPA Choosen Plaintext Attack. 58

CTA Choosen Text Attack. 58

HD Hamming Distance. 66

HE Homomorphic Encryption. 14

HW Hamming Weight. ix, 66

IoT Internet of Things. ii, 1

IOTA Internet of Things Application. 49

KPA Known Plaintext Attack. 58, 61

LIFP Large Integer Factorization Problem (LIFP). 108

LPDA Light-weight Privacy-preserving Data Aggregation. 23

ORE Order Revealing Encryption. 94

PBTree Perfect Binary Tree. 96

SHA Secure Hash Algorithm. 21

SHE Symmetric Homomorphic Encryption. 57

STSM Semi-Triangular Sparse Matrices. xii, 53

Chapter 1

Introduction

1.1 Internet of Things (IoT)

The Internet of Things (IoT) is an emerging technology that has recently attracted considerable attentions from both academia and industry. It can make a significant impact on every aspect of our daily lives, e.g., offering a wide range of services from smart home [159, 12, 70, 65, 7], smart grid [102, 33], and smart city [61, 55, 158], to smart health care [95, 115, 105, 28]. An IoT system is comprised of several functional blocks to facilitate various utilities to the system, such as sensing mechanism, communication, authentication and identification, control, and management [123, 108, 40, 94]. These functional blocks, as shown in Fig. 1.1, can be described as follows:

- Device: IoT devices are developed to provide sensing, actuation, control, and monitoring activities. They collect data from the operating environment and computing tasks on these devices can either executed locally or securely offloaded to the edge devices, fog nodes, or cloud servers. An IoT device may consist of several interfaces to capture and store the data, interact with other devices, and emit data over the network. These interfaces include I/O connectors for sensors, interfaces for Internet connectivity, memory and storage

interfaces, and audio/video interfaces.

- Communication: The communication block performs communication between devices and remote computing platforms. Although both wired and wireless components are connectivity protocols in IoT devices, they are different in capabilities, communication range, data transfer rates, power, and memory. Wireless solutions have become increasingly popular and the commonly used IoT wireless communication protocols are Bluetooth, WiFi, and Near Field Communication (NFC), while wired communication modules are mostly based on USB and Ethernet.

- Services: IoT devices can offer two classes of services, i.e., sensing and controlling, according to their function and the interaction between the sensors and the environment. In a sensing category, an IoT device can only sense and report an event to the other devices or control center. For example, air pollution sensors are deployed in the surrounding area to detect the presence of air pollution and measure its level. On the other hand, for the controlling IoT devices, not only their surrounding can be sensed, but also their parameters can be changed by feeding or triggering control signals. Thus the IoT node, in the controlling category, can change the state of a system by changing its attribute values.

- Management: Nowadays, there are numerous IoT devices with different functionalities from various vendors, which will cause interoperability issues. Due to the heterogeneous nature of IoT and its massive deployment, achieving cooperation and interoperability impose a significant burden on the developers of IoT solutions. Management block improves scalability, productivity, and efficiency in an IoT system in which numerous IoT devices with various underlying communication protocols, data formats, and technologies come together.

- Security: Although the IoT brought numerous benefits, it creates several challenges, especially in security and privacy. Addressing these challenges depends on ensuring privacy and security as a fundamental priority to guarantee IoT deployment success. Thus, the security functional block aims to secure the IoT ecosystem against cybercrimes by providing authentication, authorization, privacy, data integrity, and data security.

- Application: The application layer is the topmost layer, which interacts directly with other entities or end-users. This layer acts as an interface that provides necessary modules to control and monitor various aspects of the IoT ecosystem. It consists of smart applications from different domains in which applications allow the users to visualize and analyze the current system status or predict the upcoming events.

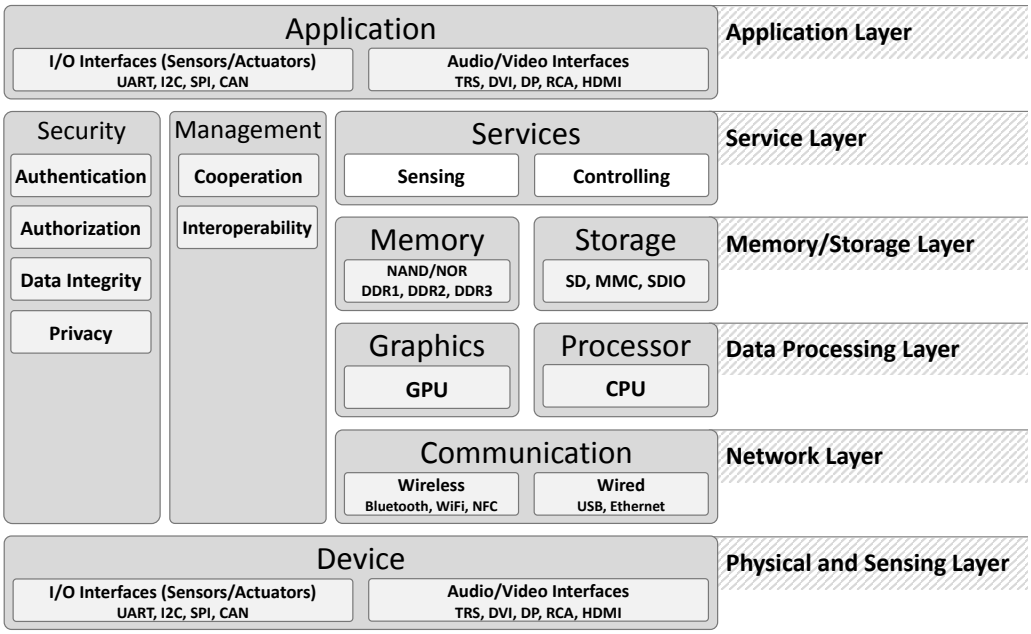


Figure 1.1: IoT functional blocks and layers

1.2 Fog computing and Fog-enhanced IoT

Despite the benefits and applications of IoT in real-world scenarios, IoT still suffers from some security, privacy, and efficiency-related shortcomings [9, 96, 107, 57]. For instance, as the number of IoT devices grows, more data are needed to be transmitted to a third party with higher frequency. Consequently, the underlying infrastructure would face big data issues in communication bandwidth and latency. Moreover, detecting innovative attacks launched by cybercriminals due to the volume, velocity, variety, and veracity of IoT big data is more difficult and the impact of the attack can be maximized. Furthermore, when huge amounts of data are being handled by the network, the security analytic report on a weekly or monthly basis would not be sufficient enough to detect and mitigate the attacks [56, 10]. To address this challenge, we can enhance IoT infrastructure with fog computing to substantially decrease the processing delay, provide real-time computation, reduce the volume of the reported data, develop privacy-preserving solutions, and offer secure networking services.

The fog computing paradigm, coined by Cisco, refers to a decentralized computing infrastructure that brings the advantages and power of cloud servers closer to where the data are being generated [36]. Fig. 1.2 illustrates the hierarchical fog computing architecture, including things layer, fog layer, and cloud/end user layer. Utilizing fog computing can bring increased security and efficiency by combining fog and IoT to form a fog-enhanced IoT platform [44, 48]. Fog-enhanced IoT is a rapidly growing technology in which many firms and industries are currently investing to develop their own real-time and low latency scenarios. Compared with the traditional IoT, fog-enhanced IoT can offer a higher level of efficiency, security features, and privacy protection thanks to the local data pre-processing, filtering, and forwarding mechanisms.

Privacy-preserving techniques are essential and challenging in fog-enhanced IoT sce-

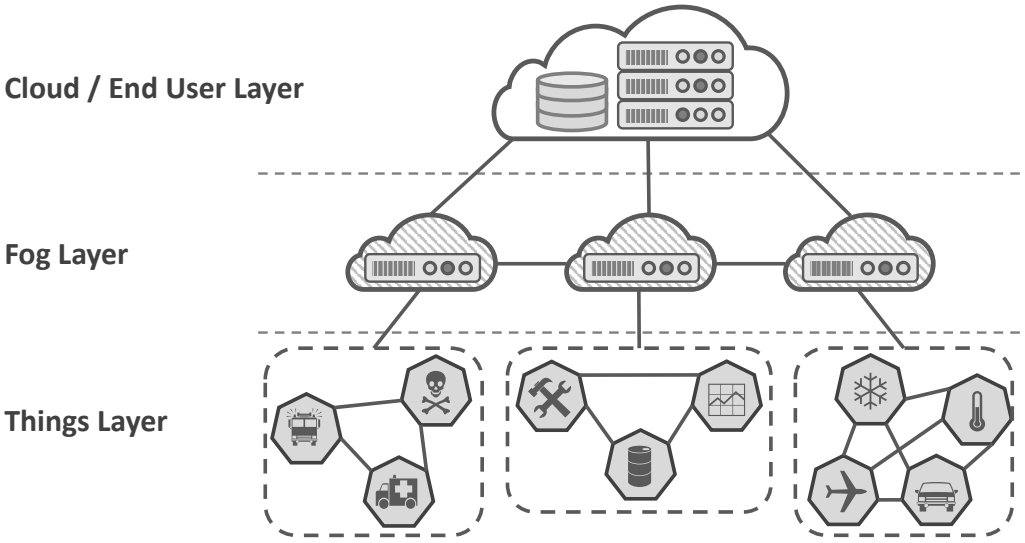


Figure 1.2: Typical fog computing architecture

narios and they have been studied in various research studies [153, 96, 98, 4, 54, 135]. The prime reason behind this is the IoT end-devices in the vicinity of the fog node. Due to this neighborhood model, fog nodes can collect sensitive data, including location, identity, and temporal information during the data transmission cycle or aggregation phase. Furthermore, since fog nodes are deployed in large areas, centralized control is becoming complex. The compromise of a poorly protected fog node can be the entry point to the network for an intruder. The intruder can violate the privacy of end-users and data.

1.3 Problem Statement

As listed in Fig. 1.3, privacy requirements in fog-enhanced IoT environments can be categorized into two main subgroups: context-based and content-based privacy [99, 121, 106, 148]. While context-based privacy focuses on the protection of the context information such as the location of the IoT devices or the time when an event is generated, content-based privacy focuses on the ability of an adversary to examine the content of the transmitted data.

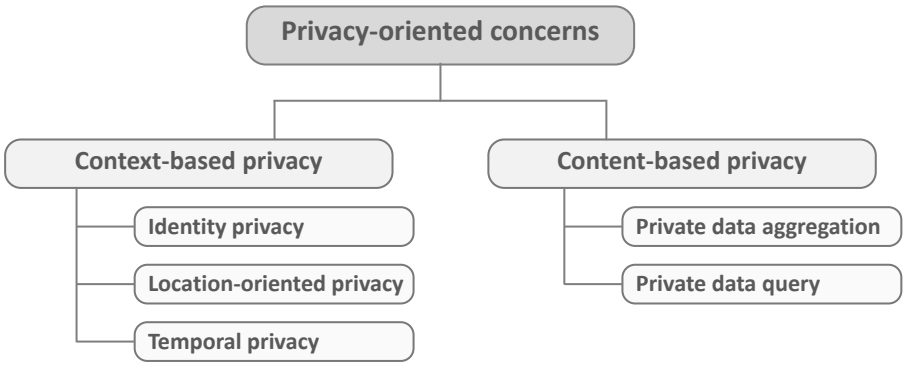


Figure 1.3: Privacy requirements in fog-enhanced IoT network

- **Context-based privacy:** Context-based privacy is related to hiding the relevant contextual information, including identity, location, and timing data referring to contextual information of the IoT devices and querying user(s) [13, 53, 103, 82]. Any leakage in the communication features, including the length and number of messages as well as the time, rate, and frequency of transmitted messages can certainly raise privacy issues.

  - Identity privacy: Managing the identity of entities in IoT environments for both IoT devices and querying users is challenging. Querying users and IoT devices want their identities to remain anonymous. Periodically updating certificates and pseudonyms to satisfy privacy concerns will lead to computational and communication overheads. Conversely, persistent identifiers ensure that a service can be offered in all steps, resulting in significant privacy concerns.

  - Location-oriented privacy: Location privacy is related to the right of IoT users to control the collection, use, and communication of personal data about their location and movements. The explosive growth of location-detection devices, e.g., GPS-enabled cell phones, have threatened the privacy of users as they require users to share private location information with the service providers. It is one of the major concerns for location-

oriented services and should be protected to avoid violating the location privacy of IoT devices and their users.

  - Temporal privacy: Temporal privacy refers to preventing an attacker from inferring the time of occurrence of an event, either the correct time or relative one.

- **Content-based privacy:** Content-based privacy aims to hide IoT sensed data such that no adversary can compromise the data value by eavesdropping or decrypting the communication between the IoT device and querying user [24, 50, 37, 161, 110, 59]. Listed as follows, we examine two main content-based privacy-preserving issues, i.e., private data query and aggregation in IoT environments.

  - Private data query: Data query privacy is related to the privacy of the queries posted to an IoT network by the querying user. It protects the IoT network and querying user from interest leakage. The leaks are not only in the querying user's interest but also in the querying user's access patterns. Therefore attaining data query privacy is becoming an increasingly important task in content-oriented privacy.

  - Private data aggregation: Data-aggregation is the process of combining information from different IoT devices as responses flow towards the querying user. Data-aggregation operators are the count, sum, average, maximum, and minimum and should be performed to reduce the communication overhead and computational cost. Privacy-preserving data aggregation allows IoT devices to simply submit their encrypted sensed data to the fog node. Then, the fog node homomorphically aggregates the encrypted data stream to preserve the privacy of the collected information, and at the same time to achieve both bandwidth and energy

efficiency.

Addressing the above challenges to achieve privacy-preserving range-aggregate queries in fog-enhanced IoT environments is attainable by encrypting the data using homomorphic cryptosystems such as the BGN [21] and Paillier [104]. Although simply encrypting the IoT end-devices' data is a straightforward solution, it incurs huge communication and computational overhead on resource-constrained IoT end-devices with highly limited storage capacity, network bandwidth, and computing power. To tackle the recent issues, fog-enhanced IoT has been proposed as an effective solution to reduce latency in responding to the queries on these resource-constrained IoT nodes. Fog-enhanced IoT, by offering a layered structure between the IoT end-devices and cloud/end-users, delivers real-time or near real-time responses and desired quality of service (QoS) level in IoT applications. Both privacy and efficiency requirements in IoT solutions lead us to consider the following scenarios of the range query scheme in fog-enhanced IoT environment in which not only the privacy of involved entities should be satisfied, but also the communication efficiency should be achieved:

1. Honest-but-curious fog node and IoT end-devices may run a profiling operation to learn as much as possible about the querying user and his/her interests, namely lower (L) and upper (U) bound values in range query $[L, U]$, where $0 \leq L \leq U \leq n-1$ and n is the maximum possible value that can be sensed by IoT devices.

2. The fog-node may be more focused and monitor the IoT end-devices to know about the prepared value ($0 \leq x_i \leq n-1$) in each IoT-device I_i.

3. The querying user may attempt to capture an individual IoT device's sensed and prepared value x_i.

4. Each IoT device I_i may track and monitor the other IoT end-devices to capture their sensed values or states.

Motivations and our major contributions for proposing an efficient privacy-preserving range-aggregate query in fog-enhanced IoT are summarized in the next sections.

1.4 Motivation

Since the fog-enhanced IoT ecosystem provides a platform that creates opportunities to query the IoT end-devices and access real-time sensed data, it is a significant driver for big data analytic and smart infrastructures. However, if poorly treated, it could disclose querying user's and IoT end-devices' privacy and their sensitive information. Therefore, the fog-enhanced IoT ecosystem and their users' privacy are ones that need to be protected from exposure. Consequently, the primary goal of the thesis is to develop a privacy-preserving range-aggregate query in this environment by maintaining control over particularly sensitive data and limit as much as possible the consequences on privacy. Moreover, at the same time, the minimum number of encrypted data packets are expected to be transferred in the query processing scheme to achieve lower latency and higher network throughput.

1.5 Major Contributions

Our thesis focuses on designing schemes to deal with the privacy-preserving issues and communication and computational challenges for range-aggregate queries in fog-enhanced IoT environments. Our major contributions can be summarized in the following points:

1. Designing decomposition/composition techniques to transform the plaintext range query $[L, U]$ into an encrypted forms that yield higher communication

and computational efficiencies. The proposed schemes accept various form of the range queries, namely, continuous($[L, U]$) and non-continuous ($[L, U] = \bigcup_{\iota=1}^{s}[L_\iota, U_\iota]$) range queries and translate them into encrypted queries to privately and securely execute queries.

2. Proposing light-weight Symmetric Homomorphic Encryption (SHE) scheme that can handle computational circuits to aggregate the IoT devices' encrypted responses in the fog node. The proposed SHE scheme performs an arbitrary number of arithmetic operations. i.e., homomorphic additions and multiplications on encrypted data. It outperforms partially homomorphic encryption schemes, allowing only one type of operation, e.g., Paillier [104], RSA [112], and El-Gamal [45]. And it also outdoes somewhat homomorphic encryption schemes, supporting an arbitrary number of additions and one multiplication, e.g., BGN [21]. Besides, it is more efficient than public-key fully homomorphic encryption schemes, such as BGV [52], BFV [47], and YASHE [23], since they are using time-consuming techniques like re-linearization and bootstrapping. Re-linearization and bootstrapping are two commonly used techniques to support more homomorphic operations and reduce the ciphertext's size, respectively.

3. Preserving the privacy of both querying user and IoT end-devices. First, no one other than the querying user can access the plaintext lower (L) and upper (U) bound values. Second, plaintext sensed data in each IoT end-device I_i (x_i) would be protected from being accessed by the fog node and the querying user.

4. Performing experiments to verify our proposed schemes' effectiveness regarding communication and computational overhead, compared to a recently published study with $O(\sqrt{n})$ communication complexity [85].

1.6 Outline of the Thesis

The remainder of this thesis is structured as follows. The background and literature review by recalling some related studies are presented in Chapter 2. Our Bloom filter-based scheme with $(n+|E|)\log n$-bit communication efficiency, where $|E|$ is the ciphertext length of Paillier cryptosystem is introduced in Chapter 3. Bloom filter data structure and Paillier homomorphic encryption are employed in the proposed decomposition/composition technique to achieve our first communication and computational efficient range-aggregate query scheme. Chapter 4 contains the detail discussion of a more efficient privacy-preserving range query scheme with $O(\log^3 n)$ communication cost. The proposed decomposition/composition technique transforms a given range query $[L, U]$ into several space-efficient semi-triangular structures, and then converts them into sparse matrices. Moreover, instead of applying existing homomorphic schemes, we devise our Symmetric Homomorphic Scheme (SHE) and the detailed analysis and homomorphic properties of SHE scheme are also presented in this chapter. Then, Chapter 5 contains our last novel decomposition/composition algorithm based on introduced reduced paths idea to achieve $O(\log^2 n)$ communication cost. This chapter is followed by communication cost discussions, security and privacy analysis, and performance evaluation. Finally, conclusions and future research work are described in Chapter 6.

Chapter 2

Literature Review

2.1 Background

Cryptography is undoubtedly the most effective technique to securely and privately transfer data between communicating parties. Encryption, decryption, and key(s) are underlying terms in cryptographic algorithms. Encryption is a process to turn the sensitive plaintext data into ciphertext that is unrecognizable and unidentifiable by an intruder when compared with original plaintext. Conversely, the decryption method allows an intended receiver to convert encrypted data back into its original form so it can be recognized. Lastly, the cryptographic key, like a password, is a secretly held string of numbers, letters, and special characters that is employed during encrypt and decrypt method call. The key can be the same for encryption and decryption methods depending on the type of encryption process. Regarding the type of key used in cryptographic schemes, cryptography techniques can be divided into two different groups - symmetric and asymmetric key encryption.

2.1.1 Symmetric Encryption

Symmetric key cryptography (or symmetric encryption) is a class of encryption schemes in which the same key is applied to encrypt and decrypt the messages (as shown in Fig. 2.1). Symmetric encryption algorithms in turn are divided into two main groups - block cipher and stream cipher schemes. Examples of popular block cipher schemes include DES, AES, Serpent, and Blowfish, and common stream cipher schemes are RC4, ChaCha, and TKIP [119, 126, 3, 3, 120].

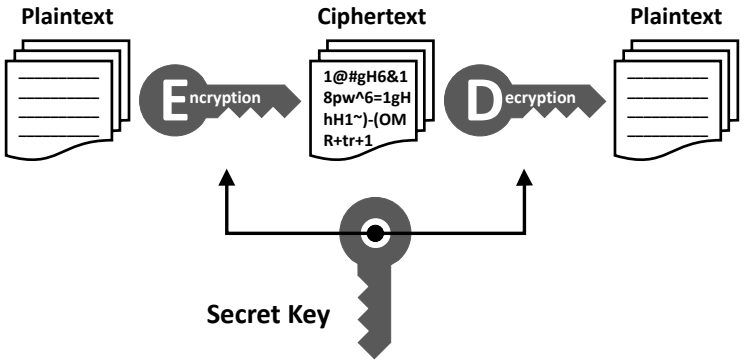


Figure 2.1: Symmetric encryption with the same key for encryption and decryption methods

2.1.2 Asymmetric Encryption

Asymmetric encryption, also known as public-key encryption, points to the cryptographic schemes that incorporate two separate keys, the public key and the private key, each of which has a specific function. As shown in Fig. 2.2, the public key is used for encryption and it can be freely published and available to anyone. In contrast, the private key should be closely protected and only viewable and accessible by the decryption function. Asymmetric encryption cryptosystems are mathematically more complex than symmetric schemes. Asymmetric schemes require more processing power for computation, and as a result, they are considerably slower than symmetric cryptosystems [3]. RSA, Diffie-Hellman, digital signature algorithm

(DSA), and elliptic curve cryptography (ECC) are examples of asymmetric cryptosystems [112, 133, 149].

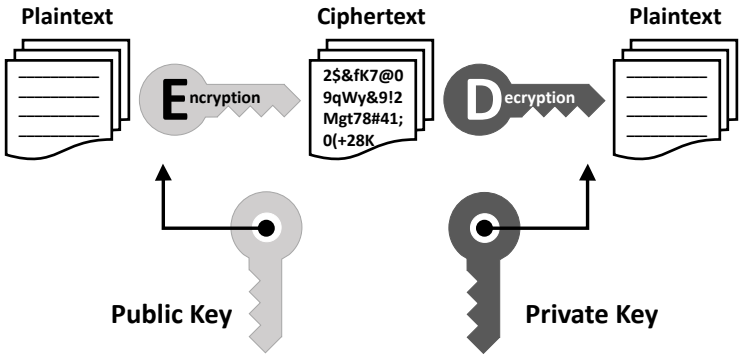


Figure 2.2: Asymmetric encryption with the public key for encryption method and private key for the decryption method

2.1.3 Homomorphic Encryption

Homomorphic encryption (HE) is a class of cryptographic schemes that allows specific types of computations to be carried out on ciphertext and obtain an encrypted result which when decrypted gives the same result as doing that computation on the original plaintext data. HE will effectively encourage the data owners to offload their storage and computation requirements to third-party infrastructures to reduce capital/operational costs. At the same time, it can ensure data privacy protection by stoping service providers from learning about users and their data. Homomorphic encryption, therefore, adequately provides the opportunity to both store and process encrypted data. HE schemes can be broadly categorized into partially, somewhat, and fully forms with respect to the number of acceptable and supported operations. Partially homomorphic encryption schemes support only either addition or multiplication function, whereas somewhat homomorphic schemes accept limited number of both operations, and finally, fully homomorphic encryption schemes will tolerate arbitrary number of operations (i.e. both multiplication and addition). For example, RSA [113] and El-Gamal [45] are multiplicative, and Paillier [104] is an additive

partially homomorphic schemes. However, BGN [21] is a somewhat homomorphic encryption that supports both addition and multiplication with subtle difference, where it accepts arbitrary number of additions and only one multiplication. During the recent decade, HE has received intense attention after the emerging of fully homomorphic encryption (FHE). It led to a major breakthrough in practical computation offloading, where the proposed FHE schemes will be able to carry out any desirable operations on ciphertexts. After almost 10 years from Gentry's proposed idea [51], the follow-up studies can be broken down into four main approaches: Ideal Lattice-based [51, 128], over Integers [132], ring learning with errors (RLWE-based) [25], and NTRU-like [83]. Although the recently suggested schemes have improved the efficiency and performance of the implementations, the overhead of the FHE schemes is still highly debatable. Thus, they cannot precisely be applied in various application scenarios and particularly in IoT solutions.

Each HE scheme will be identified by four operations: KeyGen, Enc, Dec, and Eval function which performs the function f over ciphertext arguments. Moreover, they can utilize similar symmetric or different asymmetric keys for data encryption and ciphertext decryption. In symmetric mode, both data owner and computational server are initiated by same keys whereas in asymmetric mode the private key must be kept private by data owner, and conversely the public key can be safely disseminated to cloud servers. For example, it can be seen from Fig. 2.3, homomorphic operation "$\circ$" carries out on two encrypted data $E(x)$ and $E(y)$ by only accessing public key pk, i.e., $E(m_1 \circ m_2) = E(m_1 \bullet m_2)$. Then the encrypted result $E(m_1 \bullet m_2)$ will be decrypted by private key sk, which matches the result of operations performed on the plaintext. In most homomorphic schemes "$\circ$" and "$\bullet$" are respectively referred to the common multiplication "$\times$" and addition "$+$" operations.

In the remaining part of this subsection, we will review Paillier and BGN homomorphic encryptions since these cryptosystems are extensively utilized in privacy-

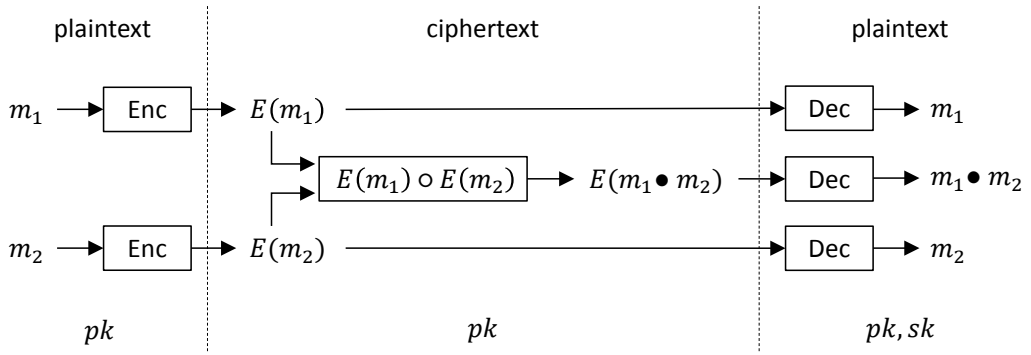


Figure 2.3: Diagram of public key homomorphic encryption (asymmetric homomorphic encryption)

preserving aggregation schemes. We summarize their descriptions and homomorphic properties.

2.1.3.1 Paillier Homomorphic Encryption

Paillier encryption is an additive homomorphic cryptosystem and is widely used in privacy-preserving applications. It has three parts, namely key generation $KeyGen(\kappa)$, encryption $E(pk, m)$ and decryption $D(sk, c)$. Table 2.1 summarizes the symbols used in Paillier scheme.

- $KeyGen(\kappa)$: Given a security parameter $\kappa \in \mathbb{Z}^+$, large prime numbers $\mathfrak{p}, \mathfrak{q}, \mathfrak{p}'$, and $\mathfrak{q}'$ are selected such that $\mathfrak{p} = 2\mathfrak{p}' + 1$, $\mathfrak{q} = 2\mathfrak{q}' + 1$ and $|\mathfrak{p}| = |\mathfrak{q}| = \kappa$. Let $\mathcal{N} = \mathfrak{p}\mathfrak{q}$ and $\lambda = lcm(\mathfrak{p} - 1, \mathfrak{q} - 1) = 2\mathfrak{p}'\mathfrak{q}'$. In addition, define a function $\mathcal{L}(x) = \frac{x-1}{\mathcal{N}}$ and randomly choose a generator $\mathfrak{g} \in \mathbb{Z}^*_{\mathcal{N}^2}$. Then, let $\mu = \mathcal{L}(\mathfrak{g}^\lambda \bmod \mathcal{N}^2)^{-1} \bmod \mathcal{N}$ to generate the public key $pk = (\mathcal{N}, \mathfrak{g})$ and the private key $sk = (\lambda, \mu)$ by calling $KeyGen(\kappa)$.

- $Enc(pk, \mathfrak{m})$: Given the public key pk and a message $\mathfrak{m} \in \mathbb{Z}_\mathcal{N}$, the message $\mathfrak{m}$ can be encrypted as $\mathfrak{c} = E(\mathfrak{m}) = \mathfrak{g}^\mathfrak{m} \cdot \mathfrak{r}^\mathcal{N} \bmod \mathcal{N}^2$, where $\mathfrak{r} \in \mathbb{Z}^*_\mathcal{N}$ is a random number.

- $Dec(sk, \mathfrak{c})$: Given the private key sk and a ciphertext $\mathfrak{c} = E(\mathfrak{m})$, the message

Table 2.1: Symbols in Paillier scheme

Symbol	Description
$\mathbb{Z}_{\mathcal{N}}$	A set of integers mod $\mathcal{N}$, $\mathbb{Z}_{\mathcal{N}} = \{0, 1, 2, \cdots, \mathcal{N}-1\}$
$\mathbb{Z}^*_{\mathcal{N}}$	The multiplicative group of invertible elements in $\mathbb{Z}_{\mathcal{N}}$, $\mathbb{Z}^*_{\mathcal{N}} = \{i \in \mathbb{Z}_{\mathcal{N}}, \exists i^{-1} \in \mathbb{Z}_{\mathcal{N}}\} = \{i \in \mathbb{Z}_{\mathcal{N}}, gcd(i, n) = 1\}$, e.g., $\mathbb{Z}^*_{12} = \{1, 5, 7, 11\}, \mathbb{Z}^*_{13} = \{1, 2, 3, 4, \cdots, 11, 12\}$
κ	Security parameter
$\mathfrak{p}$, $\mathfrak{q}$	Two large prime numbers, $\lvert\mathfrak{p}\rvert = \lvert\mathfrak{q}\rvert = \kappa$, $\mathfrak{p} = 2\mathfrak{p}' + 1$, $\mathfrak{q} = 2\mathfrak{q}' + 1$
$\mathcal{N}$	$\mathcal{N} = \mathfrak{p}\mathfrak{q}$
λ	λ is the lowest common multiple of $\mathfrak{p}-1$ and $\mathfrak{q}-1$, $\lambda = lcm(\mathfrak{p}-1, \mathfrak{q}-1) = 2\mathfrak{p}'\mathfrak{q}'$
$\mathfrak{g}$	A random integer from $\mathbb{Z}^*_{\mathcal{N}^2}$ such that $gcd(\mathcal{N}, \mathcal{L}(\mathfrak{g}^\lambda \bmod \mathcal{N}^2)) = 1$, where gcd means greatest common divisor.
$\mathfrak{m}$, $\mathfrak{c}$	A plaintext message and its corresponding ciphertext in Paillier, $0 \leq \mathfrak{m} < \mathcal{N}$
pk	Public key, $pk = (\mathcal{N}, \mathfrak{g})$
sk	Private key, $sk = (\lambda, \mu)$, $\mu = \mathcal{L}(\mathfrak{g}^\lambda \bmod \mathcal{N}^2)^{-1} \bmod \mathcal{N}$

$\mathfrak{m}$ can be recovered as $\mathfrak{m} = \mathcal{L}(\mathfrak{c}^\lambda \bmod \mathcal{N}^2) \cdot \mu \bmod \mathcal{N}$.

This partially homomorphic encryption scheme has the following homomorphic properties.

- Homomorphic Addition: Given two ciphertexts $E(\mathfrak{m}_1)$ and $E(\mathfrak{m}_2)$, we have $E(\mathfrak{m}_1) \cdot E(\mathfrak{m}_2) \rightarrow E(\mathfrak{m}_1 + \mathfrak{m}_2)$.

- Homomorphic Multiplication: Given a ciphertext $E(\mathfrak{m}_1)$ and a plaintext $\mathfrak{m}_2 \in \mathbb{Z}_{\mathcal{N}}$, we have $E(\mathfrak{m}_1)^{\mathfrak{m}_2} \rightarrow E(\mathfrak{m}_1 \cdot \mathfrak{m}_2)$.

- Self-Blindness: Given ciphertext $E(\mathfrak{m})$ and random value $\mathfrak{r} \in \mathbb{Z}^*_{\mathcal{N}}$, we have $E(\mathfrak{m}) \cdot \mathfrak{r}^{\mathcal{N}} \rightarrow E(\mathfrak{m})$. This means that the Paillier Encryption can convert a valid ciphertext into another indistinguishable yet valid one for privacy-preserving purposes.

2.1.3.2 Boneh-Goh-Nissim (BGN) Homomorphic Encryption

Unlike Paillier that is a partial homomorphic cryptosystem with supporting only one type of operations, BGN is from somewhat homomorphic encryption family. It supports an arbitrary number of additions and one multiplication by keeping the ciphertext size constant [5]. In this subsection, we briefly review this cryptosystem and its homomorphic properties. Before going into the detail of BGN, we first briefly review the bilinear pairing technique, which serve as the basis of this cryptosystem. The symbols used in BGN scheme are listed in Table 2.2.

Table 2.2: Symbols in BGN scheme

Symbol	Description
κ	Security parameter
$\mathfrak{p}$, $\mathfrak{q}$	Two large prime numbers, $\lvert\mathfrak{p}\rvert = \lvert\mathfrak{q}\rvert = \kappa$
$\mathcal{N}$	$\mathcal{N} = \mathfrak{p}\mathfrak{q}$
$\mathbb{G}$, $\mathbb{G}_T$	Two multiplicative cyclic groups of finite order $\mathcal{N}$
$\mathfrak{g}$	A generator of $\mathbb{G}$
$\mathfrak{g}$	$\mathfrak{g} \in \mathbb{G}$ is a generator of order $\mathcal{N}$
$\mathfrak{e}$	A bilinear map $\mathfrak{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$
$\mathfrak{h}$	$\mathfrak{h} = \mathfrak{g}^{\mathfrak{q}} \in \mathbb{G}_q$ is a random generator of subgroup of $\mathbb{G}$ of order $\mathfrak{p}$
$\mathfrak{m}$, $\mathfrak{c}$	A plaintext message and its corresponding ciphertext in BGN scheme, $\mathfrak{m} \in \mathbb{S} = \{0, 1, \cdots, \Delta\}$
pk	Public key, $pk = (\mathcal{N}, \mathbb{G}, \mathbb{G}_T, \mathfrak{e}, \mathfrak{g}, \mathfrak{h})$
sk	Private key, $sk = \mathfrak{p}$

Bilinear Pairing with Composite Order Let $\mathfrak{p}, \mathfrak{q}$ be two large primes of the same length, i.e., the bit-length $\lvert\mathfrak{p}\rvert = \lvert\mathfrak{q}\rvert$, and $\mathcal{N} = \mathfrak{p}\mathfrak{q}$. Two groups $(\mathbb{G}, \mathbb{G}_T)$ of the composite order $\mathcal{N}$ are called *bilinear map with composite order* if there is a computable mapping $\mathfrak{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with the following three properties: i) Bilinearity: $\mathfrak{e}(\mathfrak{g}^{\mathfrak{a}}, \mathfrak{h}^{\mathfrak{b}}) = \mathfrak{e}(\mathfrak{g}, \mathfrak{h})^{\mathfrak{a}\mathfrak{b}}$ for any $(\mathfrak{g}, \mathfrak{h}) \in \mathbb{G}^2$ and $\mathfrak{a}, \mathfrak{b} \in \mathbb{Z}_{\mathcal{N}}$; ii) Non-degeneracy: there exists $\mathfrak{g} \in \mathbb{G}$ such that $\mathfrak{e}(\mathfrak{g}, \mathfrak{g})$ is with the order $\mathcal{N}$ in $\mathbb{G}_T$; and iii) Computability: there exists an efficient algorithm to compute $\mathfrak{e}(\mathfrak{g}, \mathfrak{h}) \in \mathbb{G}_T$ for all $(\mathfrak{g}, \mathfrak{h}) \in \mathbb{G}$.

Definition 2.1.1. *(Composite Bilinear Parameter Generator) A composite bilinear*

parameter generator $\mathcal{CG}en$ is a probabilistic algorithm that takes a security parameter κ as input, and outputs a 5-tuple $(\mathcal{N}, \mathfrak{g}, \mathbb{G}, \mathbb{G}_T, \mathfrak{e})$, where $\mathcal{N} = \mathfrak{pq}$ and $\mathfrak{p}, \mathfrak{q}$ are two κ-bit prime numbers, $\mathbb{G}, \mathbb{G}_T$ are two groups with order $\mathcal{N}$, $\mathfrak{g} \in \mathbb{G}$ is a generator, and $\mathfrak{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a non-degenerated and efficiently computable bilinear map.

Let $\mathfrak{g}$ be a generator of $\mathbb{G}$, then $\mathfrak{g} = \mathfrak{g}^q \in \mathbb{G}$ can generate the subgroup $\mathbb{G}_\mathfrak{p} = \{\mathfrak{g}^0, \mathfrak{g}^1, \cdots, \mathfrak{g}^{p-1}\}$ of order $\mathfrak{p}$, and $\mathfrak{g}' = \mathfrak{g}^p \in \mathbb{G}$ can generate the subgroup $\mathbb{G}_\mathfrak{q} = \{\mathfrak{g}'^0, \mathfrak{g}'^1, \cdots, \mathfrak{g}'^{\mathfrak{q}-1}\}$ of order $\mathfrak{q}$ in $\mathbb{G}$. The SubGroup Decision (SGD) Problem in $\mathbb{G}$ is stated as follows: given a tuple $(\mathfrak{e}, \mathbb{G}, \mathbb{G}_T, \mathcal{N}, \mathfrak{h})$, where the element $\mathfrak{h}$ is drawn randomly from either $\mathbb{G}$ or subgroup $\mathbb{G}_\mathfrak{q}$, decide whether $\mathfrak{h}$ in $\mathbb{G}_\mathfrak{q}$ or not. When we assume that the SGD problem is hard, the security of the BGN homomorphic encryption can be ensured.

BGN Homomorphic Encryption BGN with more homomorphic properties consists of the following algorithms and homomorphic properties.

• *Key Generation:* Given the security parameter κ, composite bilinear parameters $(\mathcal{N}, \mathfrak{g}, \mathbb{G}, \mathbb{G}_T, \mathfrak{e})$ are generated by $\mathcal{CG}en(\kappa)$, where $\mathcal{N} = \mathfrak{pq}$ and $\mathfrak{p}, \mathfrak{q}$ are two κ-bit prime numbers, and $\mathfrak{g} \in \mathbb{G}$ is a generator of order $\mathcal{N}$. Set $\mathfrak{h} = \mathfrak{g}^\mathfrak{q}$, then $\mathfrak{h}$ is a random generator of the subgroup of $\mathbb{G}$ of order $\mathfrak{p}$. The public key is $pk = (\mathcal{N}, \mathbb{G}, \mathbb{G}_T, \mathfrak{e}, \mathfrak{g}, \mathfrak{h})$, and the corresponding private key is $sk = \mathfrak{p}$.

• *Encryption:* We assume the message space consists of integers in the set $\mathbb{S} = \{0, 1, \cdots, \Delta\}$, the size of set $\mathbb{S} = \{0, 1, \cdots, \Delta\}$ is application-oriented and much smaller than $\mathfrak{q}$, i.e., $\Delta \ll \mathfrak{q}$. To encrypt a message $\mathfrak{m} \in \mathbb{S}$, we choose a random number $\mathfrak{r} \in \mathbb{Z}_\mathcal{N}$, and compute the ciphertext $\mathfrak{c} = E(\mathfrak{m}, \mathfrak{r}) = \mathfrak{g}^\mathfrak{m}\mathfrak{h}^\mathfrak{r} \in \mathbb{G}$.

• *Decryption:* Given the ciphertext $\mathfrak{c} = E(\mathfrak{m}, \mathfrak{r}) = \mathfrak{g}^\mathfrak{m}\mathfrak{h}^\mathfrak{r} \in \mathbb{G}$, the corresponding message can be recovered by the private key $\mathfrak{p}$. Observe that $\mathfrak{c}^\mathfrak{p} = (\mathfrak{g}^\mathfrak{m}\mathfrak{h}^\mathfrak{r})^\mathfrak{p} = (\mathfrak{g}^\mathfrak{p})^\mathfrak{m}$, we can set $\hat{\mathfrak{g}} = \mathfrak{g}^\mathfrak{p}$. Then, to recover $\mathfrak{m}$, it suffices to compute the discrete log of $\mathfrak{c}^\mathfrak{p}$ base $\hat{\mathfrak{g}}$. Since $0 \leq \mathfrak{m} \leq \Delta$, the expected time is around $O(\sqrt{\Delta})$ when using the

Pollard's lambda method [93] (p.128).

The famous BGN encryption has the property of *self-blindness*, i.e., given $E(\mathfrak{m}, \mathfrak{r}) \in \mathbb{G}$, we have $E(m, r + r') \leftarrow E(m, r) \cdot h^{r'}$ is also a valid ciphertext of m. In addition, BGN also enjoys the following homomorphic properties:

- Addition in $\mathbb{G}$: Given $E(m_1, r_1) \in \mathbb{G}$ and $E(m_2, r_2) \in \mathbb{G}$, we have $E(m_1, r_1) \cdot E(m_2, r_2) = E(m_1 + m_2, r_1 + r_2) \in \mathbb{G}$. For simplicity, we omit the random items, and we have $E(m_1) \cdot E(m_2) = E(m_1 + m_2)$.

- Multiplication in $\mathbb{G}$: Given $E(m_1, r_1) \in \mathbb{G}$ and $m_2 \in \mathbb{S}$, we have $E(m_1, r_1)^{m_2} = E(m_1 \cdot m_2, r_1 \cdot m_2) \in G$. For simplicity, we have $E(m_1)^{m_2} = E(m_1 \cdot m_2)$.

- Multiplication from $\mathbb{G}$ to $\mathbb{G}_T$: Given $E(m_1), E(m_2) \in \mathbb{G}$, we have $e(E(m_1), E(m_2)) = E_T(m_1 \cdot m_2) \in \mathbb{G}_T$, where $E_T(\cdot)$ denotes a ciphertext in $\mathbb{G}_T$.

- Addition in $\mathbb{G}_T$: Given $E_T(m_1), E_T(m_2) \in \mathbb{G}_T$, we have $E_T(m_1) \cdot E_T(m_2) = E_T(m_1 + m_2)$.

- Multiplication in $\mathbb{G}_T$: Given $E_T(m_1) \in \mathbb{G}_T$ and $m_2 \in \mathbb{S}$, we have $E_T(m_1)^{m_2} = E_T(m_1 \cdot m_2)$.

2.1.4 Cryptographic Hash Function

A cryptographic hash function is a one-way mathematical algorithm applied to compress a large block of data into a much smaller fixed-length fingerprint of that data, referred to as a message digest or a hash value [62]. It can be formally expressed as $F : \{0,1\}^* \rightarrow \{0,1\}^\ell$, which accepts an arbitrary input and generates a deterministic and fixed-size (i.e., ℓ) output. This class of cryptographic functions is extensively used in a variety of security applications and network protocols, including, but not limited to, integrity verification process, digital signature generation, password verification, and proof-of-work protocol. The most common ones can be listed as MD5,

SHA-1,2,3, BLAKE2,3, and RIPEMD-160 [6].

2.1.5 Digital Signature

Digital Signature, as an authentication mechanism, is a mathematical algorithm that ensures the privacy of conversation, the integrity of data, the authenticity of digital message/sender, and non-repudiation of the sender [64]. It is based on the asymmetric encryption scheme (public-key infrastructure) and enables the sender to generate a unique code that acts as a signature for the message. To digitally sign the message, m, the sender generates the hashed version of the message. Then, this short digest, d, is encrypted by the sender's private key. Later, when the original message (m) and encrypted hash ($E(d)$) are delivered to the receiver (verifier), the receiver will generate the hash of the message m and then decrypt the encrypted hash ($E(d)$) by sender's public key to obtain d. Now, the receiver can verify the sender's signature by comparing the hash of the message and the recently decrypted hash which was generated on the sender's side. Signature is valid when hash values are equal.

2.2 Related Work

Data owners become increasingly motivated to offload their complex data collection and query processing requirements to the cloud and fog-based environments. In this section, we review the related research studies on privacy-preserving query processing schemes.

2.2.1 Naive Approach and the State-of-the-art Solution

Recently, some studies have emerged on privacy-preserving range queries in the fog-based IoT, which are closely related to our proposed schemes. In order to incorporate

the desired privacy-preserving requirements, discussed in Chapter 1, the straightforward range query scheme has an $O(n)$ communication cost. We need to first generate an encrypted array with n encrypted elements in which the entries are $E(1)$ or $E(0)$. $E(1)$ means an entry belongs to the range query, and $E(0)$ indicates an entry does not belong to the range. For example, when $n = 8$ and range query is $[L, U] = [3, 5]$, A can be formulated as $A = \{0, 0, 0, 1, 1, 1, 0, 0\}$ and the encrypted query can be considered as $E(A) = \{E(0), E(0), E(0), E(1), E(1), E(1), E(0), E(0)\}$. Then, an encrypted array $E(A)$ is sent to all IoT end-devices via the fog node. Thereupon, each IoT end-device, according to its sensed and prepared data, picks a corresponding entry from $E(A)$ and returns back its encrypted response to the fog node. After receiving the encrypted responses, the fog node homomorphically aggregates the encrypted responses to generate the final encrypted results and reports them to querying user. Finally, the querying user decrypts the ciphertexts to obtain the plaintext results. Although feasible for small n, when n becomes large, the efficiency of the proposed solution, especially communication efficiency, rapidly deteriorates. To improve this trivial solution, Lu [85] proposed a new privacy-preserving range query scheme based on a novel range query expression, decomposition, and composition technique and BGN homomorphic encryption with $O(\sqrt{n})$ communication cost. Therefore, when n becomes larger, Lu's scheme is much more efficient than the naive solution in terms of communication overhead. However, when n becomes even larger, it is more expensive in terms of both computational and communication costs. Aiming to improve the communication and computational efficiency, in this thesis, we propose three different novel decomposition/composition algorithms to build more efficient privacy-preserving range query schemes in the fog-based IoT. Our first novel range aggregate query is based on Bloom filter and Paillier homomorphic encryption. The two other schemes are on our different range decomposition/composition techniques and proposed efficient symmetric homomorphic scheme (SHE).

2.2.2 Cloud and Fog-enhanced Environments

When the privacy-preserving range query on "*the number of IoT devices whose data x_i is within the range [L, U]*" is considered as a special privacy-preserving data aggregation scheme in fog-based IoT, there are other related works close to our proposed scheme. In [86], Lu et al. addressed heterogeneous and complex data aggregation in practical IoT applications by proposing a light-weight privacy-preserving data aggregation (LPDA) for the fog-enhanced IoT environment. Their proposed scheme has been designed based on Paillier cryptosystem, the Chinese Remainder Theorem, and a one-way hash chain function to aggregate hybrid IoT end-devices' data and to early filter the injected false data at the network edge.

In [60], Huang et al. examined the fog-assisted selective aggregation operation. Precisely, they constructed a new threat model to formalize the non-collusive and collusive attacks of compromised fog nodes, and their proposed privacy-preserving and reliable selective multi-source aggregation scheme, which is comprised of BCP cryptosystem, randomized message-lock encryption, homomorphic proxy-authenticators, and multi-dimensional aggregation, can well tackle the data privacy and reliability challenges. Most recently, Mahdikhani et al. [90] proposed a privacy-preserving subset aggregation scheme in fog-enhanced IoT scenarios, which enables a querying user to gain the sum of the prepared data from a subset of IoT end-devices. The inner product similarity of the normalized vectors is securely computed to identify the subset in the querying user side and each IoT device. Only when the inner product is greater than the user's specified threshold an IoT device's data is privately aggregated to generate the final response. The dot product is one of the widely used algebraic operations that is done by multiplying the corresponding element of two vectors. However, to achieve privacy as well as efficiency multi-dot product in fog-enhanced IoT, Privacy-preserving Multi dot-product Query scheme (PMQ) was proposed by Mahdikhani and Lu [88]. PMQ utilizes BGN homomorphic encryption

to encrypt query requests and responses to satisfy privacy-preserving k dot-product query in the IoT domain, where k indicates the subset of IoT nodes that should respond to the control center. Although, due to the proposed preprocessing algorithm, the query response from the fog node to the user is communication efficient, the query request from user to the fog node can still be improved. To query specific IoT devices' data at a specific time slot in fog-based environment, PQuery [151] and XRQuery [152] proposed by Yekta and Lu. PQuery is $O(\sqrt[3]{n})$ communication overhead privacy-preserving query scheme that is formed by combining two distinctive privacy enhancing techniques, namely, private information retrieval (PIR) and oblivious transfer, to preserve the privacy in both sides, for end-user and the service provider. Unlike a prior study, XRQuery is characterized by employing a new communication-efficient PIR protocol to achieve the same privacy level as PQuery, but with more efficient communication cost, i.e., $O(\log n)$.

Privacy-preserving location-based services (LBS) queries in the cloud environment have also been widely investigated [2, 125, 163, 114, 101, 18, 164]. When a mobile user wants to query the specific entity within a range, the cloud server returns the LBS data to meet the spatial privacy-preserving range query. This privacy awareness for LBS has also been extended in fog computing platforms. Yang et al. [146] proposed a fine-grained and privacy-preserving query scheme (FGPQ) for fog computing-enhanced location-based services (LBS). Their proposed scheme employs the bilinear pairing and the asymmetric scalar-product preserving encryption algorithm (ASPE) to realize the LBS searching. In [141], Wang et al. introduced a fog sever between the mobile users and the LBS server, whereby the fog server was used instead of a traditional trusted third party to store partial information to ensure physical control. Certification, anonymity, and encryption were implemented in the fog server. Moreover, a dummy rotation algorithm has been proposed to hide the real trajectory among dummy trajectories. Based on the proposed fog comput-

ing enabled LBS framework, Liu et al. [80] suggested two low-latency LBS query schemes, i.e., kNN and $T - k$NN to support mobile queries on the Internet of Vehicles. To achieve mutual privacy preservation in both LBS provider and vehicles, oblivious transfer (OT) and ciphertext-policy attribute-based encryption (CP-ABE) have been employed. The proposed scheme minimized the latency by decreasing the required number of keys from $O(n)$ to $O(\log n)$.

Range queries are also an extensively investigated problem in database community and data structure literature. With the advent of cloud computing, secure and privacy-preserving range and range-aggregate queries, such as sum, min/max, count, average, have also become more and more popular in these years. Therefore, we will take a look at some of the recently published studies. But before doing so, we summarize them in three main categories, including tiered wireless sensor network (tiered WSN), outsourced cloud computing, and participatory sensing, to briefly review each one.

2.2.3 Tiered WSN

Privacy-preserving range query in tiered WSNs has also been broadly discussed in diverse research works as follows:

ESRQ was proposed by Zhang et al. [157] to construct an efficient and secure one-dimensional range query in two-tiered WSNs rather than range-aggregation. ESRQ utilizes the Bloom filter probabilistic data structure to encode sensed data items and queries and processes range queries according to the Bloom filter membership test. Since basic ESRQ suffers from a high false-positive ratio, they employed multiple Bloom filters technique to decrease the false-positive ratio; however, it leads to another issue (results in another problem), i.e. the higher communication cost between the sink node and master nodes.

Since ESRQ and other related studies, such as proposed BBP in [73], utilize the Bloom filter technique, they would be unsuccessful in providing precise range query results. On that account, Dong et al. [41] proposed accurate secRQ that is a secure multi-dimensional range query processing scheme in tiered WSNs. They have improved the communication cost and achieved high accuracy compared with their prior ESRQ solution in [157]. Moreover, secRQ can resist against potential security attacks in WSNs such as plaintext attacks, collusion attacks, and probability attacks. Meanwhile, in order to encode and decode the communicated data, secRQ employs the 128-bit DES symmetric-key encryption algorithm.

Chen et al. [32] proposed SQPF as a privacy and integrity preserving range query in two-tiered WSNs. They employed data hidden and verification chain techniques to, respectively, a) preserve both queries and sensory data privacy and b) ensure query result integrality. Particularly, SQPF is more efficient than its closely related design, i.e., SafeQ proposed in [30] in terms of communication cost, storage overhead and power consumption; however, its computation cost is higher than SafeQ. In November of the same year, VP^{2}RQ proposed by Dai et al. [38] is an efficient verifiable privacy-preserving range query based on the bucket partition in two-tiered WSNs. Based on the experiment results that they have provided, communication cost in VP^{2}RQ has been improved compared with existing similar protocols.

PERQ, based on data and distance comparison rules, proposed by Zeng et al.. [155] as an energy-efficient and privacy-preserving range query for multi-dimensional range query protocol in two-tiered WSNs. PERQ utilizes a stronger symmetric key algorithm, i.e. 256-bit AES, compared with their previous proposed secRQ protocol [41]. They have also addressed plaintext attacks, collusion attacks, probability attacks, and differential attacks instead of triple risky attacks in seqRQ.

Zhou et al. [162] proposed a scheme for preserving privacy query processing in cluster-based WSNs. They have applied previously proposed anonymity method and the

prefix membership verification scheme to protect the sensitive data against adversaries. Sensed data is encrypted by sensors' secret keys using a symmetric encryption algorithm, e.g., 256-bit AES, and protected by computing hash-based MAC before handing them over to the cluster-heads as an intermediate element between sensors and sink node.

2.2.4 Outsourced Cloud Computing

Computation outsourcing has become a popular alternative to hand over traditional computing tasks to cloud infrastructures; at the same time, it introduces security and privacy concerns that received extensive research attention. Here, we will review some related secure range query schemes over the cloud, including:

Samanthula et al. proposed PPRQ [117] and PRQED [118] to securely accomplish query processing tasks on outsourced Paillier-encrypted values. They employed secure comparison on encrypted integers (SEC protocol) as a basic building block to perform range queries over encrypted outsourced data in the cloud platform. The proposed SEC protocol in PRQED requires a constant number of communication rounds, in addition to being independent of the input domain size. Therefore, it demonstrates better performance than PPRQ in terms of communication and round complexities.

Rectangular range search over encrypted cloud data has been studied by P. Wang et al. [138] and B. Wang et al. [136]. The first one proposed a hierarchical encrypted index, namely R-tree, to protect data privacy and employed it to achieve secure and efficient encrypted halfspace range queries in $\mathbb{R}^d$. The second one proposed a tree-based public-key multi-dimensional range searchable encryption to attain an efficient and scalable solution on large encrypted datasets in cloud settings compared with linear search complexity.

B. Wang et al. [137] proposed two novel symmetric-key searchable encryption schemes supporting circular range queries on outsourced encrypted spatial data without revealing data privacy or query privacy to the cloud. Except knowing boolean search result and radius pattern, the cloud is not able to disclose additional privacy about data or query. Nevertheless, their solution is improved by Ren et al. [109] in terms of the efficiency of token generation, data encryption, and search processing. It leverages R-tree searchable encryption technique proposed in [138] to achieve a circular range search on outsourced spatial data. Moreover, it ensures single-dimensional privacy by disordering the elements of the query vector. In order to check the intermediate results and filter the false point, a trusted third party has also been embedded in the proposed system.

Maseedu et al. [92] proposed attribute-based range search over encrypted data in a cloud environment. They have used ciphertext-policy attribute-based encryption to satisfy both privacy and access control requirements. Moreover, their proposed scheme employed B^+-tree to create indexes and provide search efficiency and dynamic operations.

2.2.5 Participatory Sensing

Privacy concerns are becoming a major impediment in the participatory sensing (PS), where the users' sensing devices are contributing sensory information. Some related studies conducted in the PS field that focused privacy-preserving range query, including:

As participatory sensing approaches are suffering from excessive communication costs, long delays or reliance on trusted third parties, Erfani et al. in [46] proposed a data aggregation scheme for the participatory sensing system. The proposed scheme addressed user privacy and data integrity while keeping communication overhead

as low as possible. Their proposed scheme has employed an additive homomorphic stream cipher [29] to provide effective encrypted data aggregation.

Li et al. [74] proposed query and report privacy-preserving protocol (QueRe) with considering service quality to protect query privacy and report privacy in participatory sensing applications. They invented a matching process over ciphertexts to preserve privacy and guarantee service quality. Symmetric encryption, i.e., AES has been used to match queries' needs with participants' supply and asymmetric encryption, i.e., RSA to protect query privacy and report privacy.

Zeng et al. [156] used the distance comparison rule to propose a high energy-efficient and privacy-preserving (HEAP framework) range query framework, which can balance energy-efficient, privacy-preserving and data reliability in participatory sensing.

Kong et al. [66] proposed an efficient scheme that exploits structured scalars to denote the positions of the data requesters and vehicles, and achieved the privacy-preserving location matching by utilizing Paillier homomorphic cryptosystem. The proposed scheme organizes the multi-dimensional scalar into one dimension, identifies the data harvested within the target query area, and computes the aggregation results of the identified sensed data.

Lai et al. [71] proposed a privacy-preserving query processing scheme to query a range of attributes in vehicular sensor networks. In the proposed scheme, sensed data stores in local roadside units (RSU), and the global index maintains on the cloud. Submitted query to the cloud is redirected to RSUs to extract the query results.

2.3 Concluding Remarks

In this chapter, we reviewed some basic concepts, which are fundamental to provide a useful framework for this thesis, including symmetric and asymmetric encryption,

homomorphic encryption, cryptographic hash functions, and digital signature. Moreover, we introduced the commonly used homomorphic encryption cryptosystems, i.e., Paillier and BGN schemes, that are frequently referred to in this thesis. We have also discussed some related research studies in tiered WSN, outsourced cloud computing, and participatory sensing. In the following chapters, we will present several schemes, one by one, to address privacy concerns and communication and computational efficiency challenges in privacy-preserving range query in fog-enhanced IoT.

Chapter 3

Achieving Efficient and Privacy-Preserving Range Query in Fog-enhanced IoT with Bloom Filter

3.1 Introduction

Fog-enhanced Internet of Things (IoT) has attracted substantial recent studies in many fields including smart city, smart home, and eHealthcare [63]. This layered architecture consists of a few fog nodes in the upper layer that are resource-enriched entities and a large number of resource-constrained nodes (IoT devices) with limited storage and processing capabilities in the lower level. As shown in Fig. 3.1, the prepared data $D_i \in [1, n]$ in each IoT device I_i are collected and transferred to fog nodes, and the fog nodes will compute the intermediate results in order to deliver the aggregated result to the user in an efficient way. Obviously, deploying fog nodes at the network edge brings benefits such as saving resources and enhancing performance

since fog nodes can store more data and perform complex computations.However, as fog nodes are deployed at the network edge, they may not be fully trustable. Therefore, before businesses migrate to the new fog-enhanced IoT platform, they will have to be convinced that certain security level will be guaranteed while pursuing better efficiency.

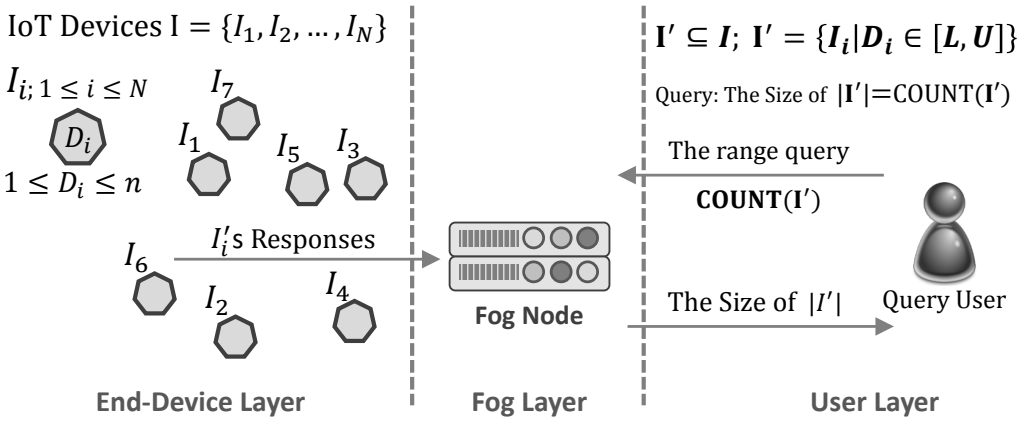


Figure 3.1: Fog-enhanced IoT architecture under consideration to support the range query for count aggregation

In this chapter, we focus on preserving privacy for the range query on the count aggregation in fog-enhanced IoT, i.e., reporting the number of IoT devices covered by the said query in a secure manner. Since IoT environments usually require reliable protection, we need to preserve not only the IoT's prepared data but also the IoT users' sensitive information, i.e. the users' usage patterns and the submitted queries. Assume that there are N IoT devices $\mathbf{I} = \{I_1, I_2, \cdots, I_N\}$ distributed over a communication network at the end-device layer with periodically prepared data D_i, where $1 \leq D_i \leq n$ and the query user at user layer who wants to know *"How many IoT devices have data lying in the range* $[L, U]$ *in* $\boldsymbol{I}$*, where* $1 \leq L \leq U \leq n$*?"*. In order to support this kind of query, in this chapter, we propose an efficient and privacy-preserving range query scheme that preserves the IoT's prepared data D_i and the user's query range $[L, U]$ by using cryptographic algorithms and space-efficient Bloom filters, while satisfying both accuracy and efficiency needs. Specifically, we employ a

novel query reformulation technique based on the cryptographic hash function, the Paillier homomorphic encryption and multiple Bloom filters to disguise all transmitted data and achieve $(n + |E_x|) \log n$-bit communication efficiency, where n is the range size as a maximum possible value that can be sensed by IoT devices and $|E_x|$ is the ciphertext length of the x cryptoscheme. More clearly, the main contributions of this work can be listed as follows:

• First, we reduce the communication cost and save more energy consumption via a novel arrangement of multiple Bloom filters which require less communication resources and speed up the aggregation process.

• Second, we preserve the confidentiality of both the user's query and the IoT devices' responses, by employing the Paillier cryptosystem and the Diffie-Hellman algorithm to protect the Bloom filters and establish the shared key between the IoT devices and the query user.

• Third, we perform thorough analysis and experiments to demonstrate the efficiency of our proposed scheme for each step from query submission to response recovery in terms of communication overhead and computational cost.

The remainder of this chapter is organized as follows. Section 3.2 introduces some preliminaries for this study. Models and design goals are described in Section 3.3. Section 3.4 elaborates the proposed scheme followed by security analysis and the detailed experimental results, respectively in Section 3.5 and Section 3.6. Related work is summarized in Section 3.7. Finally, we present our concluding remarks in Section 3.8.

3.2 Preliminaries - The Bloom Filter(BF)

In this section, we briefly review the Bloom filter data structure as one of the underlying building blocks in our proposed scheme. The other building block is Paillier

public key encryption scheme that was introduced in Chapter 2. Some used symbols in this section and chapter are first listed in Table 3.1.

Table 3.1: The list of symbols used in the proposed Bloom filter-based scheme

Symbol	Description
N	The number of the IoT devices.
I_i	The i-th IoT device in set $\mathbf{I} = \{I_1, I_2, \cdots, I_N\}$, $1 \leq i \leq N$.
n	The maximum possible value that can be sensed by IoT devices.
D_i	Sensed and prepared data in I_i range from 1 to n; $1 \leq D_i \leq n$.
L, U	Lower and upper bound in user's range query, $1 \leq L \leq U \leq n$.
I'	The subset of I where I_i's data $D_i \in [L, U]$.
R_i	I_i's response, $H(D_i \| g^{ab})$.
m	The number of inserted elements in each Bloom filter, $m = \lceil \frac{n}{\log n} \rceil$.
BF_j	The j-th Bloom filter, $1 \leq j \leq \log n$, $\forall j, \|BF_j\| = n$.
BF^0	The set of Bloom filters where $r = 0$ in corresponding $E_j(r)$.
BF^1	The set of Bloom filters where $r = 1$ in corresponding $E_j(r)$.
k	The number of shared hash functions employed by Bloom filters.
h_ℓ^{BF}	The ℓ-th shared hash function used in Bloom filters, $1 \leq \ell \leq k$.
$E_j(r)$	The corresponding encrypted value assigned to each BF_j, $r \in \{0,1\}$.
C_j	The corresponding counter assigned to each BF_j.
g	The public prime base.
a, g^a	The query private and public keys.
b, g^b	The public parameter and shared authorized key among all I_is.
g^{ab}	The shared session key between user query and all I_is.
p	The public prime modulus; $g^a \bmod p$, $g^b \bmod p$, $g^{ab} \bmod p$.
H	Cryptographic hash function, such as SHA3, BLAKE2b, and so on.

The Bloom filter, BF, is a space-efficient probabilistic data structure that is used in membership test problems [20]. An empty Bloom filter is constructed from a vector of n-bits, all initialized to zero. Then, a BF will organize a set of m elements in the above bit vector. In order to insert a new entry into BF, we need k independent hash functions, each with range $[1, n]$. For all m elements in the input set, h_ℓ^{BF}, where $1 \leq \ell \leq k$, are called to set the corresponding positions in BF to one. More precisely, each entry will pass into k uniformly distributed and independent hash function h_ℓ^{BF} to map the elements in the set to a random number over the range $[1, n]$. Finally, all the h_ℓ^{BF}-th ($1 \leq \ell \leq k$) bits in BF should be set to one. On the

other hand, to know whether an arbitrary element is in the set, one checks if all the h_ℓ^{BF}-th bits in BF are set to one. If not, the element is definitely not in the set. Otherwise, the element may be in the set. The Bloom filter is a probabilistic data structure with a false positive rate. Fortunately, the false positive rate is tunable with a clear tradeoff between the false positive rate fp and n. The probability of a false positive in this situation can be calculated as follows:

$$fp = \left(1 - (1 - \frac{1}{n})^{km}\right)^k \approx (1 - e^{-\frac{km}{n}})^k \tag{3.1}$$

As long as the false-positive rate is relatively low, the membership test result is acceptable. To minimize the false positive rate, the number of hash functions k should be around $\frac{n}{m} \ln 2$.

3.3 Models and Design Goal

In this section, we describe our system model, security model with related assumptions, and identify our design goal.

3.3.1 System Model

In our system model, we consider a typical fog-based IoT scenario that consists of a set of IoT devices $\mathbf{I} = \{I_1, I_2, \cdots, I_N\}$ arranged within the end-device layer, one fog node deployed at the fog layer, and a query user at the user layer who wishes to submit a range count queries over IoT devices, as shown in Fig. 3.1.

• IoT Devices: a set of IoT devices $\mathbf{I} = \{I_1, I_2, \cdots, I_N\}$ are deployed at the end-device layer. Each connected device $I_i \in \mathbf{I}$ in our scenario talks to other devices to accomplish the assigned tasks. They are equipped with both sensing and communication capabilities, which allows them to communicate with each other as well as

frequently collect data, like temperature, from the surrounding environment in the specific time period. In our model, each IoT device $I_i \in \mathbf{I}$ periodically reports its prepared data D_i to the fog layer *via a secure channel* which is setup between each IoT device I_i and the fog node. The secure channel prevents other IoT devices from accessing exchanged data between IoT device I_i and the fog node. Moreover, for the simplicity, we assume D_i is a positive integer in the range $[1, n]$. If the value of D_i is a floating-point value, we convert it to an integer by scaling by a power of 10.

• Fog Node: Fog node, which sits between the IoT end-devices and the user, is deployed at the fog layer to act as a gateway for IoT devices to gather the data from them, accept the queries from the user, and prepare the aggregated result for him/her. It comes with medium communication and computational capabilities, but more than IoT end-devices, to successfully execute queries.

• Query User: In our model, the query user queries the count aggregation on a set of IoT devices $\mathbf{I}' \subseteq \mathbf{I}$, where the prepared data D_i of each IoT device $I_i \in \mathbf{I}'$ falls within the range that the query user is interested in. For example, the query user can submit the query, *"How many IoT devices, whose data lies inside the range* $[L, U]$*, where* $1 \leq L \leq U \leq n$*, are in the IoT end-device layer?"*. Then, the fog node will process the range count query satisfying the range $[L, U]$ and respond the result to the user.

3.3.2 Security Model

In our security model, all entities deployed at both device and fog layers and the query user are assumed to be honest-but-curious participants, i.e., they will faithfully follow the protocols, but may be curious about the query information during data preparation and query processing steps. For instance, i) the fog node may try to read the clear prepared data D_i of IoT device I_i, ii) each IoT device may be curious about other IoT devices' data, iii) both fog node and IoT devices may try to detect the

lower and upper bound values, i.e., $[L, U]$, and finally, iv) the user may be curious about each IoT device I_i's prepared data or, at least, members of $\mathbf{I}'$. Though all entities are honest-but-curious, our security model does not consider there is any collusion among entities.

It is worth mentioning that some mature message authentication code techniques or digital signature schemes could be applied to mitigate external active attacks against data integrity and system availability. Since this study focuses on the communication efficient privacy-preserving range query for the count aggregation, those attacks are beyond the scope of this study, and would be discussed in our future work.

3.3.3 Design Goal

Based on the above system model and security model, our design goal is to propose a privacy-preserving range query for the count aggregation in the fog-enhanced IoT. Precisely, the following objectives should be achieved.

• *The proposed scheme should be privacy-preserving.* In the proposed scheme, the user's range query $[L, U]$ should be private. As the lower and upper bound values in the query $[L, U]$ expose what is of interest to the user, and an attacker can infer the user's preferences by capturing these values, therefore, during the query, no one except the user should be able to determine them. In addition, the elements of subset $\mathbf{I}'$ should be private, i.e., no one should be able to determine whether an individual IoT device, I_i, belongs to $\mathbf{I}'$ or not. Finally, the data collected by IoT devices are also sensitive and should be private. They must only be accessible by their owners. Neither the fog node nor the query user is permitted to retrieve the plaintext data D_i of I_i.

• *The proposed scheme should be efficient.* Data transmissions and security computations consume a significant portion of the energy budget, i.e., energy consumption is the prime bottleneck of IoT experiments. Therefore, we need to keep the compu-

tational overhead and communication cost among the IoT devices, fog node, and the query user as low as possible. In the proposed scheme, we achieve $(n + |E|) \log n$-bit communication efficiency between the fog node and the query user.

3.4 Our Proposed Bloom filter-based Scheme

In this section, we present our privacy-preserving range query scheme for the count aggregation in fog-enhanced IoT, which mainly consists of five parts, namely, i) system initialization, ii) query generation at query user, iii) query response at IoT devices, iv) response aggregation at the fog node; and v) result recovery at the user's side. For a better description, Fig. 3.2 outlines the detailed execution steps of the proposed scheme.

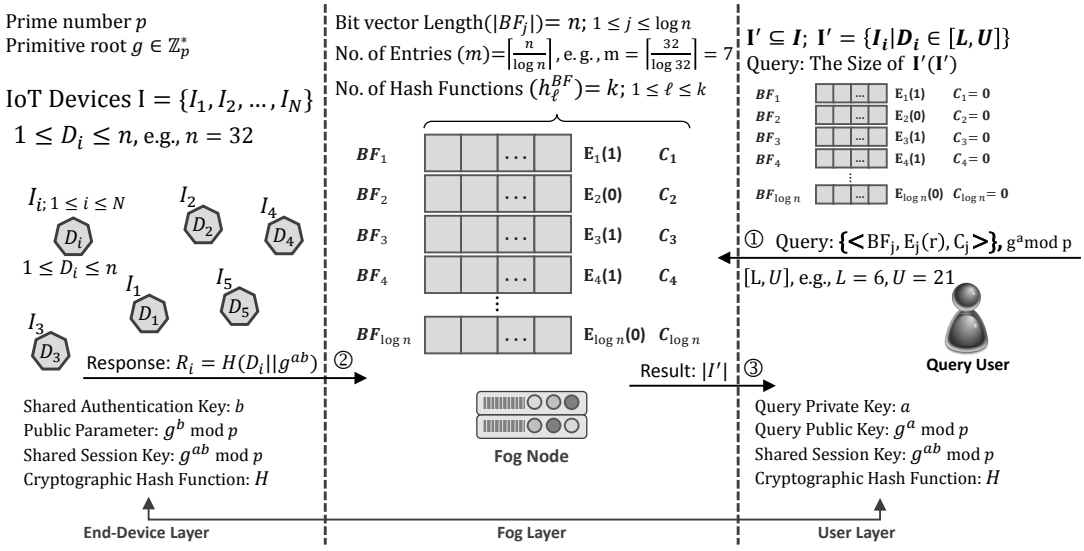


Figure 3.2: The outline of the detailed execution steps in the proposed scheme

3.4.1 System Initialization

In the system initialization phase, w.l.o.g., we assume all IoT devices and the fog node are deployed and configured by a service provider. Specifically, the service provider chooses a cryptographic hash function H, a large prime number p, and a

primitive root $g \in \mathbb{Z}_p^*$ as the public parameters. Then, the service provider chooses $b \in [1, p-1]$ as the shared authentication key for all IoT devices, computes and adds $g^b \bmod p$ into the public parameters. Meanwhile, the service provider also configures a secure channel for each IoT device and the fog node. Note that, different IoT devices have different secure channels with the fog node, and the fog node does not access the authentication key b.

3.4.2 Query Generation

When the query user launches a range query with a range $[L, U]$ for the count aggregation in fog-enhanced IoT, he will run the following steps:

- *Step 1:* The query user chooses the Paillier public-private key pair (pk, sk), selects $a \in [1, p-1]$ as the query private key, and computes the corresponding query public key $g^a \bmod p$. Also, from the public parameter $g^b \bmod p$, the query user computes the shared session key $g^{ab} \bmod p$.

- *Step 2:* The query user defines the query as a set of triple $< BF_j, E_j(r), C_j >$, where BF_j, $(1 \leq j \leq \log n)$, is the j-th Bloom Filter, which is labelled with a Paillier ciphertext $E_j(r)$, $r \in \{0, 1\}$, together with a count C_j with the initial value $C_j = 0$. Concretely, each Bloom filter BF_j consists of a vector of n bits and will store no more than $m = \lceil \frac{n}{\log n} \rceil$ items from the whole range $[1, n]$ to achieve the probability of false positive $fp \approx (1 - e^{-\frac{k}{\log n}})^k \approx e^{-\ln n \ln 2} = n^{-\ln 2}$ as $k = \frac{n}{m} \ln 2$. Moreover, based on the value of $E_j(r)$, all bloom filters can be divided into two categories: BF^1 is a set of bloom filters labelled with $E(1)$, and BF^0 is a set of bloom filters labelled with $E(0)$ as follows:

$$\begin{cases} BF^1 = \{< BF_j, E_j(r), C_j > | r = 1\}; \\ BF^0 = \{< BF_j, E_j(r), C_j > | r = 0\}. \end{cases} \tag{3.2}$$

- *Step 3:* To generate the query, the whole data range $[1, n]$ will be organized into the Bloom filters ($BF^0 \cup BF^1$), where the items in the query range $[L, U]$ will be inserted into BF^1 and the remaining items ($[1, n] - [L, U]$) will put into BF^0. Note that, for each $i \in [1, n]$, we do not insert the original value i, but its keyed hash value $H_i = H(i||g^{ab})$, into bloom filters, which can prevent the fog node for some guessing attack. The details of our algorithm can be found in Algorithm 1. Note that, the method PickRandomBloomFilter() in Algorithm 1 picks a random bloom filter BF_t and increment the internally assigned counter. When the counter reaches m, PickRandomBloomFilter() will ignore that Bloom filter from the next round of selection.

- *Step 4:* The query user sends the query $< BF_j, E_j(r), C_j >$, $1 \leq j \leq \log n$, and $pk, g^a \bmod p$ to the fog node. Upon receiving the query request, the fog node forwards $g^a \bmod p$ to all IoT devices.

Algorithm 1 Query Generation

Input: n, the query range $[L, U]$, empty Bloom filters BF_j
Output: Set $\{< BF_j, E_j(r), C_j >\}$

for each $BF_j \in \{< BF_j, E_j(r), C_j >\}$ **do** # $1 \leq j \leq \log n$
  $C_j \leftarrow 0$
  $r \leftarrow$ RandomInetger$(0, 1)$ # $r \in \{0, 1\}$
  $E_j \leftarrow$ Paillier.E(r) # **E**$(\cdot)$: The Paillier encryption method.
  if $r = 0$ **then**
    $BF^0 \leftarrow BF^0 \cup BF_j$
  else
    $BF^1 \leftarrow BF^1 \cup BF_j$
for each $i \in [1, n]$ **do**
  $H_i \leftarrow H(i||g^{ab})$
  if $i \in [L, U]$ **then**
    $BF_t \leftarrow$ PickRandomBloomFilter(BF^1)
  else
    $BF_t \leftarrow$ PickRandomBloomFilter(BF^0)
  BF_t.AddElement(H_i) # Call all $h_\ell^{BF}(H_i)$ to add the key.
return Set $\{< BF_j, E_j(r), C_j >\}$

3.4.3 IoT Devices' Response

For each IoT device $I_i \in \mathbf{I}$, after receiving $g^a \bmod p$ from the fog node, it will run the following two steps:

- *Step 1:* I_i uses the shared authenticated key b and $g^a \bmod p$ to compute the shared query session key $g^{ab} \bmod p$.
- *Step 2:* I_i computes the keyed hash value of its prepared data $D_i \in [1, n]$, i.e., $R_i \leftarrow H(D_i||g^{ab})$, and returns R_i as a response to the fog node via a secure channel.

3.4.4 Response Aggregation at Fog Node

Upon receiving each R_i from IoT device I_i, $1 \leq i \leq N$, the fog device starts the membership query process to find the possible BF_j. If R_i belongs to BF_j, the corresponding counter C_j for each BF_j is incremented. After that, Paillier homomorphic methods are called to calculate the final encrypted result *ERes*, and *ERes* will be returned back to the query user. The details are described in Algorithm 2.

3.4.5 Result Recovery at User Side

Once the encrypted result *ERes* is received, the query user can use the Paillier private key *sk* to recover the query result *Res*. The correctness is as follows.

$$\begin{aligned} ERes &= \prod_{j=1}^{\log n} \left(E_j(r)^{C_j} \right)_{r \in \{0,1\}} \\ &= \prod_{[L,U] \in BF_j} \left(E_j(1)^{C_j} \right) \cdot \prod_{[L,U] \notin BF_j} \left(E_j(0)^{C_j} \right) \\ &= E\left(\sum_{[L,U] \in BF_j} 1.C_j + \sum_{[L,U] \notin BF_j} 0.C_j \right) \\ &= E\left(\sum_{[L,U] \in BF_j} C_j \right) \xrightarrow{\text{Decrypt by } sk} \sum_{[L,U] \in BF_j} C_j = Res = |\mathbf{I}'| \end{aligned} \tag{3.3}$$

Algorithm 2 Fog node response

Input: R_i, # The IoT devices's response.
Output: Encrypted count result *ERes*.
1: **for each** R_i **do** # R_i is I_i's response and $1 \le i \le N$.
2: **for each** BF_j **do** # $1 \le j \le \log n$
3: **if** BF_j.MembershipTest(R_i) **then**
4: $C_j \leftarrow C_j + 1$
5: $ERes \leftarrow$ Paillier.E(0) # Encrypted result *ERes*.
6: **for each** BF_j **do** # $1 \le j \le \log n$
7: $EC_j \leftarrow$ Paillier.Mul(E_j, C_j) # Ej^{C_j}.
8: $ERes \leftarrow$ Paillier.Add($ERes, EC_j$) # $\sum_{j=1}^{\log n} E_j.C_j$.
9: $ERes \leftarrow$ Paillier.SelfBlind($ERes, r$) # $r \in \mathbb{Z}_n^*$
10: **return** *ERes*

3.5 Security Analysis

In this section, we analyze the security of the proposed scheme, especially focusing on the privacy-preserving aspects, i.e., the privacy of the range count query $[L, U]$ as well as the privacy of the subset $\mathbf{I}'$.

• *The query range* $[L, U]$ *is privacy-preserving in the proposed scheme.* In order to achieve $(n + |E|) \log n$-bit communication efficiency, the query range $[L, U]$ in the proposed scheme is formulated by $\log n$ multiple Bloom filters $BF^0 \cup BF^1$. As IoT devices know the session key $g^{ab} \bmod p$, each IoT device can determine which bloom filter that the hash value $H(i||g^{ab})$ of i, $1 \leq i \leq n$, falls into. Nevertheless, we will show (by contradiction) that each IoT device still cannot determine the query range $[L, U]$. Suppose an IoT device can determine the query range $[L, U]$, it means that the IoT device can also distinguish the bloom filters BF^1 labelled with $E(1)$ from the bloom filters BF^0 labelled with $E(0)$, which however will contradict with the fact that Paillier encryption is semantically secure. Therefore, IoT device cannot determine the query range $[L, U]$. Likewise, the fog node without knowing the session key $g^{ab} \bmod p$ also has no idea on the query range $[L, U]$. As a result, the query range $[L, U]$ is privacy-preserving in the proposed scheme.

• *The subset* $\boldsymbol{I'}$ *is also privacy-preserving in the proposed scheme.* Without knowing the session key $g^{ab} \bmod p$, the fog node cannot recover the IoT data D_i from its keyed hash value $H(D_i||g^{ab})$. At the same time, though the fog node can know the hash value $H(D_i||g^{ab})$ falls into a bloom filter BF_j, the fog node cannot determine whether $BF_j \in BF^1$. Therefore, each IoT device I_i's data D_i and whether or not I_i belongs to $\mathbf{I}'$ are privacy-preserving. Note that, since a secure channel is employed between each IoT device and the fog node, other IoT devices also cannot know I_i's data D_i. From Algorithm 2, the fog node can know each C_j, $j \in [1, \log n]$. However, the fog node cannot determine $E_j(r)$ is a ciphertext of 0 or 1. Therefore, in eye of the fog node, there are $\sum_{i=1}^{\log n} \binom{\log n}{i} = 2^{\log n} - 1 = n - 1$ possible values of $Res = |\mathbf{I}'|$. (We assume the query range $[L, U]$ exists in at least one bloom filter.) As a result, the fog node can guess the correct $|\mathbf{I}'|$ only with a probability $1/(n-1)$. From the above analysis, we can conclude that the subset $\mathbf{I}'$ is also privacy-preserving in the proposed scheme.

3.6 Performance Evaluation

Table 3.2: The parameter settings

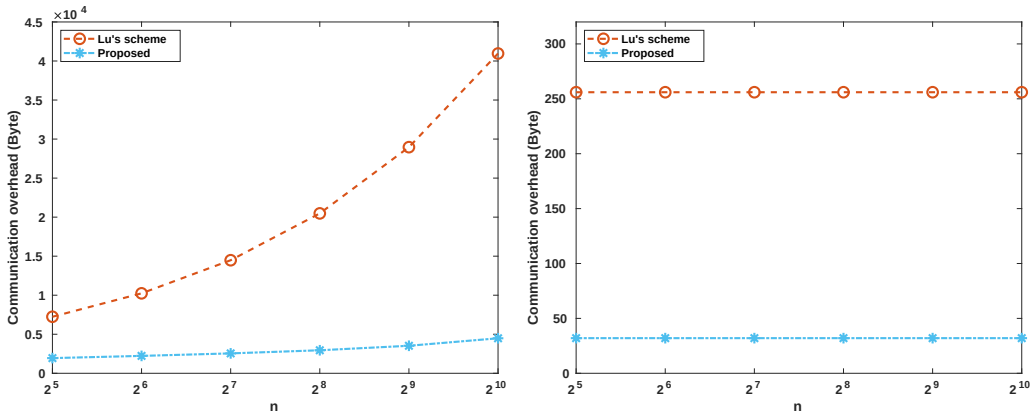
Paramter	Value
κ	$\kappa = 512, \lvert\mathfrak{p}\rvert = \lvert\mathfrak{q}\rvert = \kappa, \mathcal{N} = \mathfrak{pq}$
$\lvert E_x\rvert$	Ciphertext length in BGN and Paillier, $\lvert E_{BGN}\rvert$, $\lvert E_{Paillier}\rvert$
p	Diffie-Hellman's public prime modulus, $\lvert p\rvert = 2048$
n	The upper bound of [1,n], $n = \{2^5, 2^6, 2^7, 2^8, 2^9, 2^{10}\}$
H	Cryptographic hash function, SHA3-256.
k	Number of Bloom filters' hash functions, $k = 3$
N	Number of IoT devices, $N = 1000$

In this section, we evaluate the performance of our proposed scheme in terms of communication and computational costs. To this end, each processing step of our proposed scheme is compared with the corresponding step in the most recently BGN-based scheme (denoted as Lu's scheme) in [85]. The platform to compare them is an Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz, with 8 GB main memory, running Linux operating system (Ubuntu 16.04), and using Java (JDK 1.8). The detailed parameter setting is listed in Table 3.2.

3.6.1 Communication Overhead

In this subsection, we evaluate the communication cost in both directions between the query user and the fog node and then the fog node and the IoT devices. Fig. 3.3(a) compares the communication overhead of transferring the query from user to the fog node in both schemes with respect to n varying from 2^5 to 2^{10}. It can clearly be seen that the communication overhead in Lu's scheme increases rapidly, while in our proposed scheme, it is remarkably low and highly efficient. Because the communication cost of Lu's scheme arises from $5\sqrt{n} \times |E_{BGN}|$, whereas in our proposed scheme comes from $n \log n$-bits multiple Bloom filters and $\log n \times (|E_{Paillier}| + |C_j|)$-bits for both corresponding Paillier ciphertexts and counter parameters. It should

be noted that our scheme is highly efficient when the range size is less than 2^{18}, i.e., $1 \leq n \leq 2^{18}$. Moreover, the comparison of the communication cost between an individual IoT device and the fog node in both schemes is depicted in Fig. 3.3(b). While in our proposed scheme the cryptographic hash function's output is submitted to the fog node, in Lu's scheme the BGN ciphertext is responsible. Consequently, our scheme with SHA3-256 hash function is eight times faster.



(a) Requests from the query user to the fog node

(b) Responses from an IoT device to the fog node

Figure 3.3: Communication overhead comparisons (linear scale) between the proposed scheme and Lu's scheme with respect to n

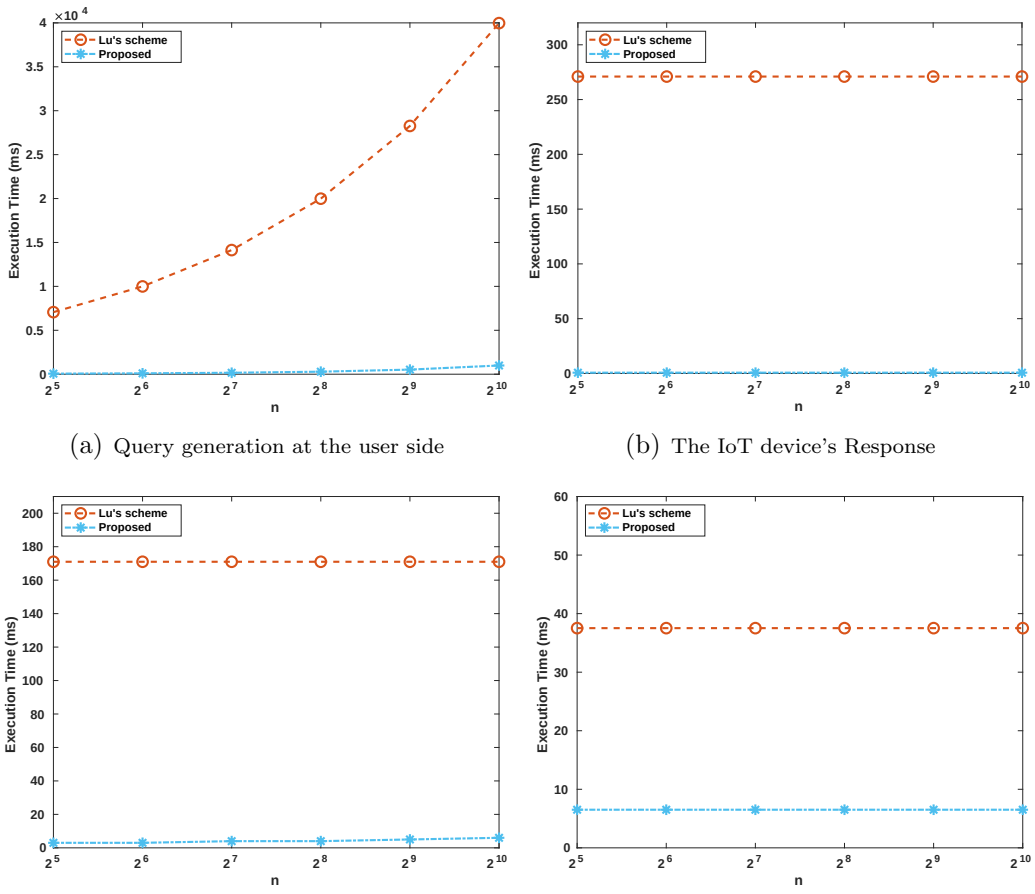
3.6.2 Computational Costs

In Fig. 3.4, we compare the average computational cost in each step between our proposed scheme and Lu's. Fig. 3.4(a) depicts the computational cost of the user's query generation in both schemes. To simplify analysis and comparison, we look for a dominant operation, i.e., the encryption method and count the number of times that the method is called. In our scheme, we assign an encrypted value to each Bloom filter, therefore the Paillier encryption method is called $\log n$ times, whereas in Lu's scheme, the BGN encryption method is called $5\sqrt{n}$ times. Therefore, our solution with $\log n$ encryption outperforms this step in terms of time and bandwidth consumption. Note that, the average encryption time, under the same κ value,

in Paillier is also much faster than BGN, e.g., in our reference machine, Paillier encryption takes about 6.9 ms whereas BNG encryption takes about 249.8 ms on average. Fig. 3.4(b) presents the computational cost of the IoT device's response and it is obviously independent of n. Since generating the IoT device's response in Lu's scheme requires time-consuming homomorphic operations, i.e., three homomorphic additions, three homomorphic multiplications, and one conversion, it is slow in comparison to our scheme (less than a millisecond), because a single simple function call suffices to generate the needed cryptographic hash value. Fig. 3.4(c) depicts the case for the response aggregation at the fog node. We know the aggregation in Lu's scheme requires N BGN-homomorphic additions, whereas in our scheme, as listed in Algorithm 2, requires $\log n$ Paillier-homomorphic multiplications, $(\log n) - 1$ Paillier-homomorphic addition, and $N \log n$ times calls to k different Bloom filter hash functions. Therefore, Lu's scheme has constant time, but ours varies with n. Finally, Fig. 3.4(d) indicates the response decryption time at query user. From the figure, we can see our scheme has a shorter execution time due to the faster decryption time from the Paillier cryptosystem.

3.7 Related Work

Privacy-preserving range queries have been investigated widely in different platforms, such as wireless sensor networks [155, 162] and outsourced cloud computing[131, 144, 79]. They have also been studied recently in fog-enhanced IoT applications [88, 152, 85]. Since our proposed scheme is an extension to improve the communication and computational overhead of the proposed work in [85], we focus and review it briefly. As the most related work, Lu's recently proposed scheme [85] develops a novel decomposition technique to establish $O(\sqrt{n})$ communication efficiency. It employs BGN homomorphic cryptosystem to preserve the privacy of the IoT devices' data and the



(a) Query generation at the user side

(b) The IoT device's Response

(c) Aggregate response at the fog node

(d) Response recover at the user side

Figure 3.4: Computational cost comparisons between proposed scheme and Lu's scheme

user's query. A simple contiguous range query of length n reorganizes by a query expression, decomposition, and composition technique into five BGN-encrypted vectors of length $\sqrt{n}$ to achieve the aforesaid efficiency gain. Different from the Lu's scheme, our proposed scheme can support composite non-contagious range queries by employing multiple enabled Bloom filters and Paillier cryptosystem to achieve $(n + |E|) \log n$-bit communication efficiency and better computational performance.

3.8 Concluding Remarks

In this chapter, we proposed a communication-efficient privacy-preserving range query for the count aggregation scheme in fog-enhanced IoT deployments. The proposed scheme employs the Paillier homomorphic encryption, an elegant Bloom filter data structure to securely exchanges queries, responses, as well as intermediate results to prevent any information leakage. Although probabilistic Bloom filters have been employed, we have effectively organized and enabled them with corresponding Paillier encrypted values to achieve accurate results and $(n+|E|)\log n$-bit communication efficiency. In our next chapter, we will endeavor to reduce the communication cost within the tiers by proposing a new decompostion/composition technique to achieve $O(\log^3 n)$ communication cost.

Chapter 4

Achieving $O(\log^3 n)$ Communication-Efficient Privacy-Preserving Range Query in Fog-Based IoT

4.1 Introduction

With ongoing advances in research and development of the Internet of Things (IoT), the connected IoT devices are deployed in an ever-increasing number of projects in different fields [11, 19, 77, 134, 42, 97], varying from cryptocurrencies (IOTA Tangle [100, 8, 124]) to food safety industries [43, 81, 27] and smart grid [142, 31, 15], and they generate huge volumes of data. As reported in [130], there will be more than 10 billion IoT devices interconnected via the internet by 2020, and all these devices will generate around 4.4 trillion GB of data. As a result, IoT emerging technologies will be expected to investigate in the decades ahead, as they have been broadly studied over the past few years. In conventional IoT applications, IoT devices are

employed to gather distributed data from the operational environment and transmit them to central decision-making points in the cloud. Obviously, in outsourcing time-sensitive big data applications, some potential concerns such as response latency, bandwidth usage, and reliability should be carefully considered[87]. To this end, fog-based IoT has been emerged to address the factors mentioned above, where the big data is being analyzed [150, 127, 22, 58]. The response is aggregated closer to where the IoT devices are deployed [35, 9, 88, 91, 68]. At the same time, the fog extends the conventional cloud to the edge of the network providing real-time or at least fast enough computation and efficient storage and networking services between the IoT devices and cloud layer/users[39, 35, 1]. However, since the fog nodes are deployed at the network edge, they may not be fully trusted. As a result, privacy-preserving techniques are essential in fog-based IoT scenarios[153, 96, 98, 4, 54, 135]. In this chapter, we will devise an efficient privacy-preserving range query scheme in fog-based IoT. In our system, there are N IoT devices $\{I_i|1 \leq i \leq N\}$, and each I_i reports its data $x_i \in [0, n-1]$, where n is the maximum value of the sensed data. The querying user submits his/her range query request to a fog node at the network edge. The range query is in the form of $[L, U]$, which means that the user queries "*the number of IoT devices whose data x_i is within the range [L,U], where* $0 \leq L \leq U \leq n-1$". A potential application scenario for this kind of range query is a smart grid, where a smart grid operator uses the query to track the finer-grained electricity consumption of a neighborhood for better scheduling and optimization purposes [85]. Aiming to achieve desired privacy goals, the querying user expects to keep the range values, i.e., L and U, confidential, and each I_i tries to keep its individual data x_i secure. A naive approach towards such a privacy-preserving range query is to use homomorphic encryption (HE) schemes such as BGN [21] and Paillier [104]. Suppose that $n = 10$, and there are $N = 20$ IoT devices $\{I_i|1 \leq i \leq 20\}$, and each I_i is preparing its data x_i, where $0 \leq x_i \leq n-1 = 9$. At the same time, the user

intends to run the range query with corresponding parameters $L = 2$, and $U = 6$. With the naive solution, the user first represents the query range $[L, U]$ as an array $A = \{0, 0, 1, 1, 1, 1, 1, 0, 0, 0\}$ of size n. $A[i]$ is set to one if $L \leq i \leq U$ and otherwise, it is set to zero. Then, he/she encrypts A with the HE encryption algorithm $Enc()$ as $Enc(A) = \{Enc(A[0]), Enc(A[1]), \cdots, Enc(A[n-1=9])\}$. After that, he/she sends $Enc(A)$ to the fog node, which forwards $Enc(A)$ to each IoT device. Upon receiving $Enc(A)$ from the fog node, each IoT device I_i with its ready-to-report value x_i calculates $c_i \leftarrow \text{selfBlind}(Enc(A[x_i]))$ and sends it to the fog node. Note that, the *selfBlind* is a nice property of HE for better privacy preservation, where a ciphertext can be transformed into another one without changing the corresponding plaintext [84, 140, 143]. Finally, the fog node aggregates the ciphertexts received from the IoT devices into a single ciphertext, i.e., $C = \sum_{i=1}^{N} c_i$ and then sends C to the querying user. Although the above naive solution can satisfy the privacy concerns in the range query, its communication cost is prohibitively high, i.e., $O(n)$. To reduce the communication cost, Lu [85] proposed a privacy-preserving range query scheme with $O(\sqrt{n})$ communication cost, which is acceptable for small values of n but it is not efficient for large n. Therefore, designing a communication efficient privacy-preserving range query scheme in fog-based IoT is still challenging, especially for a large value of n. In this research, we propose a new efficient privacy-preserving range query scheme with $O(\log^3 n)$ communication cost for fog-based IoT. More precisely, our main contributions in this research are summarized as follows:

- First, we design a novel decomposition technique to transform a given range query $[L, U]$ into several space-efficient semi-triangular structures, and then converts them into sparse matrices for preserving query privacy.

- Second, we propose a Symmetric Homomorphic Encryption (SHE) scheme that can support different number of additive and multiplicative homomorphic operations. For example, we can setup the scheme in a way to support b homo-

morphic multiplication, where $b = \log_2 n$. SHE scheme not only offers different homomorphic operations, but also provides an efficient and secure way to prepare, disseminate, and calculate ciphertexts.

- Third, based on the decomposition technique and the proposed symmetric homomorphic scheme, we present an efficient privacy-preserving range query scheme with $O(\log^3 n)$ communication cost for fog-based IoT.

- Finally, we analyze the security of our proposed scheme and evaluate its performance. The results show that our proposed scheme can indeed satisfy the desired privacy concerns, and outperforms the most recently developed approach in [85] in terms of computational costs and communication overhead.

The remainder of the chapter is structured as follows. The system and security models along with design goal are introduced in Section 4.2. Section 4.3 contains the description of our symmetric homomorphic encryption scheme and its properties. Then, Section 4.4 presents our proposed scheme, followed by security analysis in Section 4.5. Furthermore, the detailed performance evaluation is organized in Section 4.6. Some related works are discussed in Section 4.7 and, finally, concluding remarks are made in Section 4.8.

4.2 Models and Design Goal

In this chapter, we descibe our system model and security model, and describe the design goal for our communication-efficient privacy-preserving range query. For clear description, some used symbols are first listed in Table 4.1.

Table 4.1: The list of symbols used in the proposed Semi-Triangular Sparse Matrices (STSM) scheme

Symbol	Description
N	The number of the IoT devices.
I_i	The i-th IoT device in set $\mathbf{I} = \{I_1, I_2, \cdots, I_N\}$, $1 \leq i \leq N$.
n	The maximum possible value that can be sensed by IoT devices.
x_i	Sensed and prepared data in I_i range from 1 to n; $0 \leq x_i \leq n-1$.
L, U	Lower and upper bound in range query, $0 < L = 2^a \leq U = 2^{a'} < n-1$.
$\mathbf{I}'$	The subset of I where I_i's data $x_i \in [L, U]$.
PP	Public parameter in SHE scheme; $PP = (k_0, k_1, k_2, \mathcal{N})$.
SK	Secure key in SHE scheme; $SK = (p, q, \mathcal{L})$.
$Enc(m)$	Encryption method in cryptosystem, $m \in \mathcal{M} \in \{0,1\}^{k_1}$.
$Dec(c)$	Decryption method in cryptosystem, $\lvert c\rvert = \lvert\mathcal{N}\rvert = 2k_0$.
B	Range's binary representation, $\lvert B\rvert = U - L + 1$.
S	Hamming weight classification of B, $\lvert S\rvert = \log_2 n + 1$=b+1.
C	Cumulative sum values for each entry in S, $\lvert C\rvert = \lvert S\rvert$.
R_i	Semi-triangular matrix generated from each C_i.
T_i	The corresponding sparse matrix for each R_i.

4.2.1 System Model

Fig. 4.1 illustrates the proposed system model from a high-level perspective. It consists of three major entities, namely IoT devices $\mathbf{I} = \{I_1, I_2, \cdots, I_N\}$, a fog node in the network edge and a querying user.

• IoT devices $\mathbf{I} = \{I_1, I_2, \cdots, I_N\}$: A myriad of IoT devices $\mathbf{I}$ of size N are remotely spread across the device layer and get involved in a specific IoT solution. Each IoT device I_i gathers raw data x_i from the physical sensing environment and will typically perform some processing tasks on them. The prepared data x_i afterwards will be reported to the fog node in the upper layer for later processing. It is worth noting that, in our system model, the reported data will be converted into an integer in the range of $[0, n-1]$ to clarify the processing steps and declare the scheme. To this end, we can easily transform floating-point values into integers by applying simple scaling, truncating or rounding functions. However, using a common conversion method(s) or a combination of them depends on what kind of conversion is desired

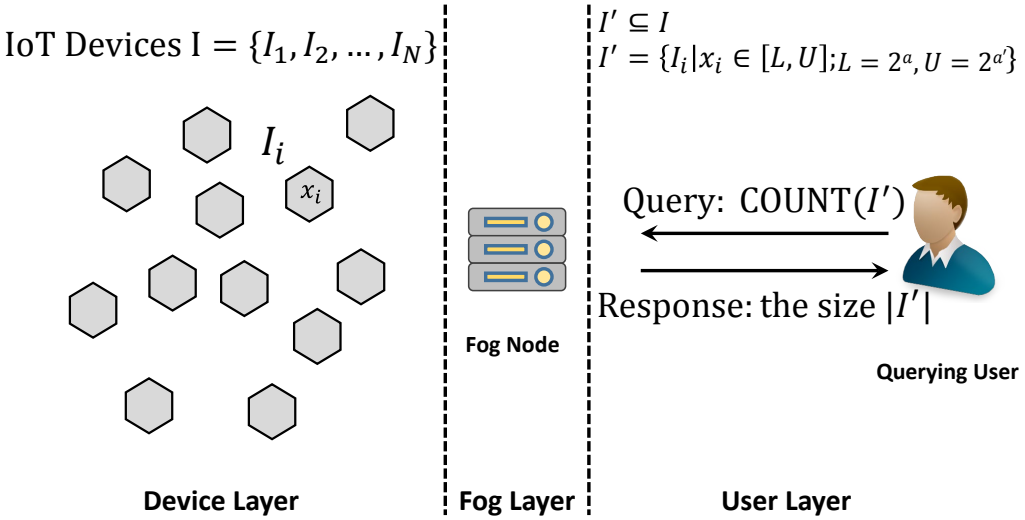


Figure 4.1: System model under consideration

and what works best, e.g., $x_i = 198.73549$ can be transformed into an integer value by $truncate(x_i = 198.73549 * 10) = 1987$ if the upper bound value for n is $2^{11} = 2048$; and $x_i = 1987354$ for a scaling factor of 10000 where $n = 2^{21} = 2097152$. Therefore, accepting large scaling factors and consequently large n values, which our model supports, will lead to more accurate results. More precisely, our proposed scheme would be more effective for large n values, i.e., $n \geq 2^{17}$, and much more efficient than a recent related study in [85].

• Fog node: The fog layer, as an intermediate layer, comprises various communication and computing elements, namely fog nodes, which handle a near real-time stream of requests and responses. In our system model, the fog node is a computational entity more powerful than IoT devices but not as powerful as cloud entities. After receiving prepared data x_i from lower layer IoT device I_i, the fog node applies filtering rules and performs aggregation operations. Afterwards, the aggregated data will be stored in local storage or rerouted to the upper layer for further analysis. According to her/his query preference, a querying user in the user layer will submit a range query to the fog layer and shortly receive the reply from the fog node.

• Querying user: As depicted in Fig 4.1, a querying user can submit a secure range

query to fetch the number of IoT devices whose prepared data x are within the range $[L, U]$, where L and U are, respectively, the lower and upper bounds of the range, and are both in the form of "two to the power of a positive integer", i.e., $L = 2^a \leq U = 2^{a'}$. Suppose that $\mathbf{I}'$ is a subset of $\mathbf{I} = \{I_1, I_2, \cdots, I_N\}$, where the prepared data x_i of each IoT device $I_i \in \mathbf{I}'$ is within the range $[L, U]$, i.e.,

$$\mathbf{I}' = \{I_i | I_i \in \mathbf{I} \ \wedge x_i \in [L, U]; L = 2^a, U = 2^{a'}\} \tag{4.1}$$

The querying user will then submit the following range count query to find the number of IoT devices whose x_i is within the given range, i.e.,

$$\text{Query: COUNT}(\mathbf{I}') \longrightarrow \text{Response: the size of } \mathbf{I}' \ (|\mathbf{I}'|).$$

4.2.2 Security Model

In our security model, all entities deployed at both devices and fog layers, are assumed to be honest-but-curious participants, i.e., they are obliged to follow the protocols faithfully but may be curious about the query information during data preparation and throughout the query processing steps. For example, i) the fog node may try to identify the IoT devices I_i with exact data x_i, ii) each IoT device may be curious about other IoT devices' data, iii) both fog node and IoT devices may try to detect the lower (L) and/or upper (U) bound values, and finally, iv) the user may be curious about each IoT device I_i's prepared data or, at least, prepared data in each IoT device in $\mathbf{I}'$. Note that the honest-but-curious assumption would be guaranteed in practice, since the service providers should protect their own reputation and financial interests. In addition, there would be no collusion between entities, such as sharing their results. Lastly, to mitigate external active attacks, some mature message authentication code techniques or digital signature schemes could be applied. However, considering

such potential attacks are beyond the scope of this research and we focus on the communication efficient privacy-preserving range query.

4.2.3 Design Goal

Our goal is to have an efficient privacy-preserving range query scheme with $O(\log^3 n)$ communication cost in a fog-based IoT environment based on the requirements above in both the system model and security model. Specifically, the following two objectives should be satisfied.

• *The proposed scheme should be privacy-preserving.* In the proposed scheme, the user's query range $[L, U]$ should be privacy-preserving, i.e., no one, except the user, can determine $[L, U]$. In addition, the elements of subset $\mathbf{I}'$ should also be privacy-preserving, i.e., no one can determine whether a specific IoT device belongs to $\mathbf{I}'$ or not, and only the querying user can know COUNT($\mathbf{I}'$) after the range query.

• *The proposed scheme should be communication efficient.* To address the above privacy objectives, additional communication overhead will be incurred in the range query, as discussed previously in Section 4.1. Therefore, in the proposed scheme, we attempt to improve communication efficiency by achieving $O(\log^3 n)$ communication cost, which is better than the current most communication efficient scheme with $O(\sqrt{n})$ communication cost in [85].

4.3 Proposed Symmetric Homomorphic Encryption

To build our communication-efficient privacy-preserving range query scheme, we need to adopt homomorphic encryption techniques, which can support both homomorphic addition and multiplication. However, most existing homomorphic encryption schemes use public key settings, which are not computationally efficient. For this

reason, in this chapter, we first introduce an efficient symmetric homomorphic encryption (SHE) scheme, which will serve as the fundamental building block for our privacy-preserving range query.

4.3.1 Description of SHE Scheme

As a symmetric homomorphic encryption scheme, SHE mainly consists of three algorithms: *key generation*, *encryption*, and *decryption*. In the following, we describe them in detail.

- *Key Generation:* Given the security parameters (k_0, k_1, k_2) satisfying $k_1 \ll k_2 < \frac{k_0}{2}$, generate the secret key $SK = (p, q, \mathcal{L})$, where p, q are two large prime numbers with $|p| = |q| = k_0$ and $\mathcal{L}$ is a random number with the bit length $|\mathcal{L}| = k_2$. Then, compute $\mathcal{N} = pq$ and set the public parameter $PP = (k_0, k_1, k_2, \mathcal{N})$. At the same time, set the message space $\mathcal{M}$ as $\{0,1\}^{k_1}$.

- *Encryption:* A message $m \in \mathcal{M}$ can be encrypted with the secret key $SK = (p, q, \mathcal{L})$ as

$$c = Enc(m) = (r\mathcal{L} + m)(1 + r'p) \bmod \mathcal{N} \tag{4.2}$$

  where $r \in \{0,1\}^{k_2}$ and $r' \in \{0,1\}^{k_0}$ are two random numbers.

- *Decryption:* A ciphertext $c = Enc(m)$ can be decrypted with the secret key $SK = (p, q, \mathcal{L})$ as

$$Dec(c): \ m = (c \bmod p) \bmod \mathcal{L} \tag{4.3}$$

The correctness of the decryption is as follows.

$$\begin{aligned} Dec(c) &= (c \bmod p) \bmod \mathcal{L} \\ &= (((r\mathcal{L} + m)(1 + r'p) \bmod \mathcal{N}) \bmod p) \bmod \mathcal{L} \\ &= (r\mathcal{L} + m) \bmod \mathcal{L} \quad (\because 2k_2 < k_0) \\ &= m \quad (\because k_1 \ll k_2) \end{aligned}$$

Security of SHE. Obviously, our SHE scheme is simple and efficient. Next, we will prove that it is also secure. Before that, we introduce different types of attack models in which the cryptanalyst knows or chooses plaintexts (m_i) or ciphertexts (c_i) and attempts to break an encrypted message. Dependent on the type of information available to the cryptanalyst, common cryptanalysis methods can be classified into the ciphertext-only attack (COA), the known-plaintext attack (KPA), chosen-plaintext attack(CPA), chosen-ciphertext attack (CCA), and chosen-text attack (CTA) [129].

- *Ciphertext-only attack (COA):* In this case, the cryptanalyst has access only to the ciphertexts (c_i). Hence, this is the most difficult attack scenario from the cryptoanalytic point of view. Furthermore, since the cryptanalyst has essentially nothing to do except analyzing ciphertexts, this is the weakest kind of attack model.

- *Known-plaintext attack (KPA):* The cryptanalyst is given a pair or limited pairs of plaintexts and the corresponding ciphertexts, i.e., (m_i, c_i), which he did not choose and cannot produce more pairs. The goal is to find the key or decrypt the rest of the data for which plaintext is unknown.

- *Chosen-plaintext attack (CPA):* In this attack scenario, the cryptanalyst is able to choose any plaintext and obtain the corresponding ciphertext. This allows him to query the encryption oracle generating a ciphertext of a desired

plaintext. Typically, it does not seem a practical possible scenario. But in the case of a public key cryptosystem where the encryption key is public, the cryptanalyst can encrypt any number of messages. Therefore certain attacks are possible using CPA in which the attacker has multiple messages under the same key and can retrieve the encryption key. Ciphertext indistinguishability as a standard tool is a property in many encryption schemes in which an intruder will be unable to distinguish pairs of ciphertexts based on the message they encrypt. Indistinguishability under chosen-plaintext attack (IND-CPA) is equivalent to the property of *semantic security*. A cryptosystem is considered secure in terms of indistinguishability if the success probability of any probabilistic polynomial time adversary has only a negligible advantage over random guessing. Therefore the adversary cannot identify the message with a probability significantly better than that of random guessing $\frac{1}{2}$.

- *Chosen-ciphertext attack (CCA):* This is the opposite of the previous attack, in which the cryptanalyst can choose any ciphertext he wants and obtain the corresponding plaintext using a decryption oracle; however, he does not know the actual decryption key.

- *Chosen-text attack (CTA):* The CTA occurs when the cryptanalyst uses a combination of chosen plaintext and chosen ciphertext. It is assumed that cryptanalyst has access to both the encryption and decryption oracle. Therefore, he can query encryption/decryption oracle for any given arbitrary plaintext and fetch corresponding ciphertext and vice versa. The goal of the cryptanalyst is to find the secret key, then all information is available to him.

Let $M = r\mathcal{L} + m$, we know $M < p$, and then the ciphertext $c = Enc(m) = (r\mathcal{L} + m)(1 + r'p) \bmod \mathcal{N}$ becomes

$$c = M \cdot (1 + r'p) \bmod \mathcal{N} \tag{4.4}$$

Theorem 4.3.1. *Given the ciphertext* $c = M \cdot (1+r'p) \bmod \mathcal{N}$*, obtaining the message* M *is equivalent to factoring the larger integer* $\mathcal{N} = pq$*.*

Proof. First, if $\mathcal{N} = pq$ can be factored into p and q, from the decryption algorithm, we can use p to obtain $M = c \bmod p$. We know $M = r\mathcal{L}+m < p$, then by factorizing $\mathcal{N} = pq$, we have p and q, then

$$\begin{aligned} c &= M \cdot (1 + r'p) \bmod \mathcal{N} \\ &= M \cdot (1 + r'p) + k\mathcal{N} \\ &= M \cdot (1 + r'p) + kpq \quad \text{for some integer } k \end{aligned}$$

Now M can be recovered from c:

$$\begin{aligned} c \bmod p &= \Big(M \cdot (1 + r'p) + kpq\Big) \bmod p \\ &= \Big(M + Mr'p + kpq\Big) \bmod p \\ &= M \quad (\because M = r\mathcal{L} + m < p) \end{aligned}$$

Second, if we can obtain M from $c = M \cdot (1 + r'p) \bmod \mathcal{N}$, then the large integer $\mathcal{N} = pq$ is factored. In the ciphertext $c = M \cdot (1+r'p) \bmod \mathcal{N}$, the value of $r'p$ could be less $\mathcal{N}$ or greater than or equal to $\mathcal{N}$. We assume $\Delta = r'p \bmod \mathcal{N}$. Obviously $\Delta < \mathcal{N}$.

- Case 1: if $r'p < \mathcal{N}$, we have $\Delta = r'p < \mathcal{N}$.
- Case 2: if $r'p \geq \mathcal{N}$, we have $r'p = \Delta + k\mathcal{N} = \Delta + kpq$ for some integer k:

$$r'p = \Delta + kpq \rightarrow \Delta = r'p - kpq \xrightarrow{p|r'p-kpq} p|\Delta$$

From $p|\Delta$, we have $\Delta = \delta \cdot p < \mathcal{N}$ for some integer δ.

Then, $c = M \cdot (1 + r'p) \bmod \mathcal{N} = M \cdot (1 + \Delta) \bmod \mathcal{N}$. If M can be recovered from c, we have $\Delta = (c \cdot M^{-1} - 1) \bmod \mathcal{N} < \mathcal{N}$, then we can find $p = \gcd(\Delta, \mathcal{N})$, and then large integer $\mathcal{N} = pq$ is factored.

Therefore, obtaining the message M from c is equivalent to factoring the larger integer $\mathcal{N} = pq$. Only if the larger integer factoring problem is hard with proper parameter settings, we can make sure that the message M is secure. □

Theorem 4.3.2. *Our proposed SHE scheme is secure under the known-plaintext attack (KPA).*

Proof. Given a pair or pairs of (c_i, m_i), a ciphertext $c_i = Enc(m_i) = (r_i\mathcal{L} + m_i)(1 + r_i'p) \bmod \mathcal{N}$ and its corresponding plaintext m_i, we will show that an adversary still cannot obtain the secret key $SK = (p, q, \mathcal{L})$. First, in order to obtain the secret key (p, q), according to Theorem 4.3.1, it is equivalent to obtaining the message $M_i = r_i\mathcal{L} + m_i$ from (c_i, m_i). Since the bit-length of the whole $r_i\mathcal{L}$ is $|r_i\mathcal{L}| = 2k_2$, we know the adversary can guess the correct $r_i\mathcal{L}$ and $M_i = r_i\mathcal{L}+m_i$ only with probability $\frac{1}{2^{2k_2}}$. Therefore, only if the parameter k_2 is well set, the secret key (p, q) cannot be obtained from (c_i, m_i). Second, the adversary can directly guess the correct secret key $\mathcal{L}$ with probability $\frac{1}{2^{k_2}}$. But he still needs to guess the correct r_i with probability $\frac{1}{2^{k_2}}$ to guess the correct $r_i\mathcal{L}$ and $M_i = r_i\mathcal{L} + m_i$. Therefore, guessing $\mathcal{L}$ with probability $\frac{1}{2^{k_2}}$ does not help guess M and the secret key (p, q) with probability $\frac{1}{2^{2k_2}}$. As a result, our proposed SHE scheme is secure under the known-plaintext attack (KPA). □

Note that in the proposed system model throughout this thesis, none of the participated entities, except the querying user, have access to neither the secret parameters (key) nor encryption oracle. Therefore, considering the known-plaintext attack can assure the security of the proposed system model. However, the proposed SHE scheme can provide a stronger security guarantee, i.e., secure against

chosen-plaintext attack as discussed and proved in Zheng et al [160].

4.3.2 Homomorphic Properties of SHE

Given the public parameter PP, our proposed SHE scheme has the following homomorphic properties:

- *Homomorphic Addition-I:* Given two ciphertexts $c_1 = Enc(m_1) = (r_1\mathcal{L} + m_1)(1 + r'_1 p) \bmod \mathcal{N}$, $c_2 = Enc(m_2) = (r_2\mathcal{L} + m_2)(1 + r'_2 p) \bmod \mathcal{N}$, we have $c_1 + c_2 \rightarrow Enc(m_1 + m_2)$. This is because

$$\begin{aligned} & c_1 + c_2 \bmod \mathcal{N} \\ &= (r_1\mathcal{L} + m_1)(1 + r'_1 p) + (r_2\mathcal{L} + m_2)(1 + r'_2 p) \bmod \mathcal{N} \\ &= (r_1 + r_2)\mathcal{L} + m_1 + m_2 + \alpha \cdot p \bmod \mathcal{N} \\ &\Rightarrow ((c_1 + c_2) \bmod p) \bmod \mathcal{L} \\ &= ((r_1 + r_2)\mathcal{L} + m_1 + m_2) \bmod \mathcal{L} = Enc(m_1 + m_2) \end{aligned}$$

  where $\alpha = (r_1 r'_1 + r_2 r'_2)\mathcal{L} + (m_1 r'_1 + m_2 r'_2)$.

- *Homomorphic Multiplication-I:* $c_1 \cdot c_2 \rightarrow Enc(m_1 \cdot m_2)$. This is because

$$\begin{aligned} & c_1 \cdot c_2 \bmod \mathcal{N} \\ &= (r_1\mathcal{L} + m_1)(1 + r'_1 p) \cdot (r_2\mathcal{L} + m_2)(1 + r'_2 p) \bmod \mathcal{N} \\ &= \beta'\mathcal{L} + m_1 \cdot m_2 + \alpha' \cdot p \bmod \mathcal{N} \\ &\Rightarrow ((c_1 \cdot c_2) \bmod p) \bmod \mathcal{L} \\ &= (\beta'\mathcal{L} + m_1 \cdot m_2) \bmod \mathcal{L} = Enc(m_1 \cdot m_2) \end{aligned}$$

  where α' and β' are derived from $c_1 \cdot c_2$.

- *Homomorphic Addition-II:* Given a ciphertext $c_1 = Enc(m_1) = (r_1\mathcal{L} + m_1)(1 + r'_1 p) \bmod \mathcal{N}$, and a plaintext m_2, we have $c_1 + m_2 \rightarrow Enc(m_1 + m_2)$. This is

because

$$
\begin{aligned}
& c_1 + m_2 \bmod \mathcal{N} \\
&= (r_1\mathcal{L} + m_1)(1 + r_1'p) + m \bmod \mathcal{N} \\
&= r_1\mathcal{L} + m_1 + m_2 + (r_1\mathcal{L} + m_1) \cdot p \bmod \mathcal{N} \\
&\Rightarrow ((c_1 + m_2) \bmod p) \bmod \mathcal{L} \\
&= (r_1\mathcal{L} + m_1 + m_2) \bmod \mathcal{L} = Enc(m_1 + m_2)
\end{aligned}
$$

- *Homomorphic Multiplication-II:* $c_1 \cdot m_2 \rightarrow Enc(m_1 \cdot m_2)$. This is because

$$
\begin{aligned}
& c_1 \cdot m_2 \bmod \mathcal{N} \\
&= (r_1\mathcal{L} + m_1)(1 + r_1'p) \cdot m_2 \bmod \mathcal{N} \\
&= r_1 m_2\mathcal{L} + m_1 \cdot m_2 + (r_1\mathcal{L} + m_1)m_2 \cdot p \bmod \mathcal{N} \\
&\Rightarrow ((c_1 \cdot c_2) \bmod p) \bmod \mathcal{L} \\
&= (r_1 m_2\mathcal{L} + m_1 \cdot m_2) \bmod \mathcal{L} = Enc(m_1 \cdot m_2)
\end{aligned}
$$

Based on the above homomorphic properties of SHE, we will design our communication efficient privacy-preserving range query scheme in the next section. The SHE source code in Java, is included in Appendix A.

4.4 Our Proposed Scheme

In this chapter, we will describe our privacy-preserving range query scheme in fog-based IoT. Before delving into the details, we first present our decomposition technique, which is a building block in our proposed scheme.

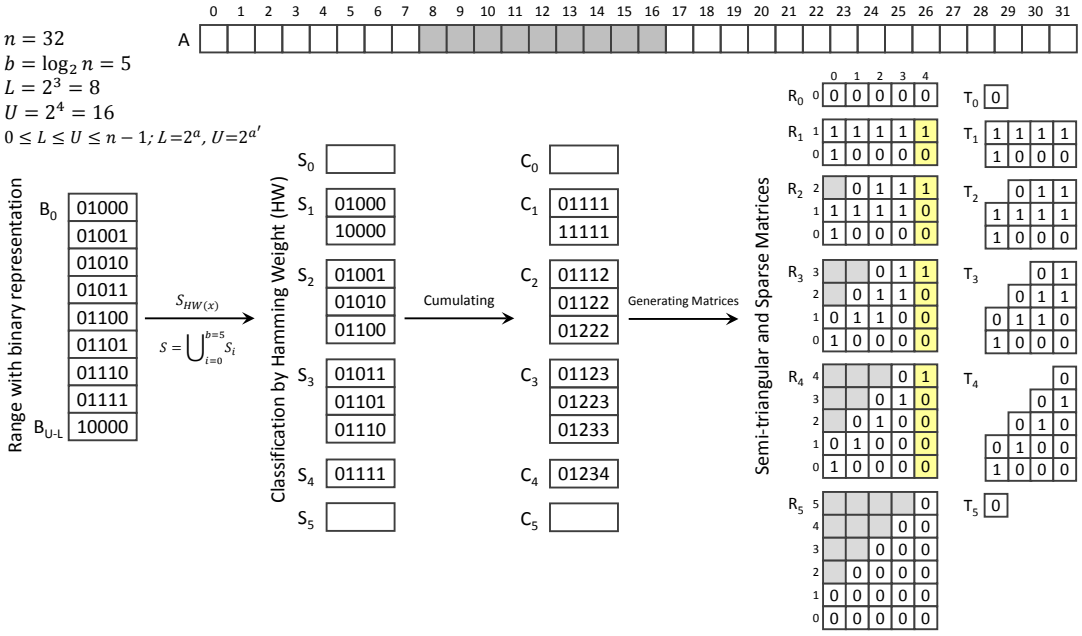


Figure 4.2: Converting a range query into semi-triangular and sparse matrices. For example, the corresponding range query $[L, U] = [2^3, 2^4]$ of array A where $L \leq U \leq 31$ is transformed into $(\log_2 n) + 1 = 6$ sparse matrices $T_0, T_1, \cdots, T_{\log_2 n}$.

4.4.1 The Decomposition Technique

Our decomposition technique can transform a query range $[L = 2^a, U = 2^{a'}]$ into $(\log_2 n + 1)$ sparse matrices, where $a \leq a'$ and $0 \leq L \leq U \leq n - 1$ and consists of five steps, 1) range query formulation, 2) binary representation of the range, 3) Hamming-weight classification, 4) cumulative sums structures construction, and finally 5) semi-triangular matrices generation. Note that, for clear description, we also take the values $n = 32$, $[L = 2^3, U = 2^4]$ and instantiate a simple example in Fig. 4.2, in which 6 sparse matrices $\{T_0, \cdots, T_5\}$ are generated, since $(\log_2 n) + 1 = (\log_2 32) + 1 = 6$.

4.4.1.1 Range Query Formulation

Given the lower bound L and upper bound U, the general syntax of a range query statement can be outlined using an array $A[0..n-1]$ as follows,

$$A[i] = \begin{cases} 1, & L \leq i \leq U; \\ 0, & \text{otherwise}, \end{cases} \tag{4.5}$$

where $0 \leq i \leq n-1$.

4.4.1.2 Binary Representation of the Range

In this step, the given range query $[L, U]$ is converted into a vector of bit-strings. As shown in Algorithm 3, for each $i \in [L, U]$, generate the equivalent binary sequence of i by simply applying two bitwise operations, *RightShift* and *AND*. Clearly, the expected output B is a vector of binary strings where $|B| = U - L + 1$ and each entry B_ℓ, where $0 \leq \ell \leq U - L$, of B is a string of zeros and ones of length b, i.e., $B_\ell \in \{0,1\}^b$ and $b = \log_2 n$, as shown in Fig. 4.2.

Algorithm 3 Range's Binary-Representation

Input: $b = \log_2 n$, L, and U; $0 < L = 2^a \leq U = 2^{a'} < n - 1$
Output: Vector $<B_>$; $|B| = U - L + 1$, and $|B_\ell \in B| = b$

```
1: B ← Ø
2: ℓ ← 0                                  # 0 ≤ ℓ ≤ U − L
3: for each i ∈ [L, U] do
4:     c ← b − 1
5:     B_ℓ ← ""
6:     while c ≥ 0 do                     # & is bitwise AND operator
7:         B_ℓ.Append(RightShift(i, c) & 1)
8:         c ← c − 1
9:     ℓ ← ℓ + 1
10: return B
```

4.4.1.3 Hamming Weight (HW) Classification

The Hamming weight of a given bit-string x is the Hamming distance (HD) of two equal-length bit-strings x and zero-codewords, i.e., $HD(x, \{0\}^{|x|})$. In other words, the Hamming weight of a given bit-string x is the number of ones in x, and it can be computed by the HammingWeight() procedure in Algorithm 4. For example, the Hamming weight of B_0 in Fig. 4.2, can be denoted as $HD(\text{“01000”}, \text{“00000”}) = 1$. Therefore, all the input bit-strings in vector B will be categorized into $\log_2 n + 1$ classes to form vector S as shown in Algorithm 4, as well as $S = (S_0, S_1, \cdots, S_5)$ in our simple example in Fig. 4.2, which will be used to generate the cumulative sum structure in the next step.

Algorithm 4 Hamming Weight Classification

Input: Vector $<B>$
Output: Vector $<$Vector $<S>>$ and $|S| = \log_2 n + 1 = b + 1$

1: $S \leftarrow \emptyset$
2: **for each** $B_\ell \in B$ **do**
3: $\quad \boldsymbol{\omega} \leftarrow$ HammingWeight(B_ℓ) # $0 \leq \boldsymbol{\omega} \leq b$
4: $\quad S_{\boldsymbol{\omega}}$.Add$(B_\ell)$
5: **return** S

6: **procedure** HammingWeight(x) # Only iterates as many times as there are ones in the x
7: $\quad t \leftarrow 0$
8: $\quad$ **while** $x \neq 0$ **do**
9: $\quad\quad x = x$ & $(x - 1)$ # & is bitwise AND operator
10: $\quad\quad t \leftarrow t + 1$
11: $\quad$ **return** t

4.4.1.4 Cumulative Sum Structure Construction

As shown in Algorithm 5, in this step, our goal is to generate the cumulative sum structure for each entry in S, i.e., $\forall s_j \in S_i, 0 \leq i \leq b$. As seen in lines 6-11 of the algorithm, a cumulative sum t is a same-length sequence of partial sums of a given bit sequence x or clearly the value at each index of t equals the sum of current and

all the preceding values. For example, as shown in Fig. 4.2, the cumulative sum $c_0 \in C_3$ = "0, 1, 1, 2, 3" for $s_0 \in S_3$ = "01011" can be interpreted as: $c_{00} = s_{00} = 0, c_{01} = s_{01} + c_{00} = 1 + 0 = 1, c_{02} = s_{02} + c_{01} = 0 + 1 = 1, c_{03} = s_{03} + c_{02} = 1 + 1 = 2$, and finally $c_{04} = s_{04} + c_{03} = 1 + 2 = 3$. It should be noted that, unlike the B's and S's elements, each $c_j \in C_i$ is itself a vector of integer values instead of bits to store the partial sums.

Algorithm 5 Cumulative Sum Structure

Input: $S = \{S_i\} = \big\{\{s_j | \textsc{HammingWeight}(s_j) = i\}\big\}$
Output: Vector < Vector <Vector $<C>$>> and $|C| = |S|$

1: $C \leftarrow \emptyset$
2: **for each** $S_i \in S$ **do** # $0 \le i \le b$, $0 \le j \le$ No of entries in S_i
3: **for each** $s_j \in S_i$ **do**
4: $C_{ij}.\textsc{Add}\Big(\textsc{CummulativeSum}(s_j)\Big)$
5: **return** C

6: **procedure** $\textsc{CummulativeSum}(x)$
7: Vector $<t>$ $\leftarrow \emptyset$
8: $t.\textsc{Add}(x_0)$
9: **for** $i = 1$ to $\textsc{Length}(x) - 1$ **do**
10: $t.\textsc{Add}\big(x_i + t.\textsc{Get}(i-1)\big)$
11: **return** t

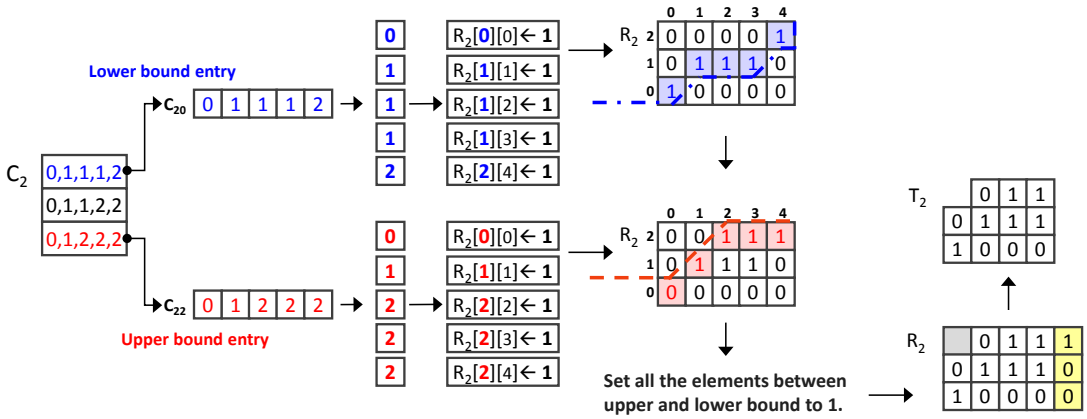


Figure 4.3: Generating matrix R_2 and sparse structure T_2 from cumulative sum C_2

4.4.1.5 Semi-triangular Matrices Generation

Finally, Algorithm 6 converts the cumulative sum structure C into semi-triangular matrices. For clarity, we take the cumulative sum C_2 in Fig. 4.2 as an example to show how to transform C_2 to a semi-triangular matrix R_2, as depicted in Fig. 4.3. Before analyzing the example, it should be noted that the result matrix R_i has $i+1$ rows and $b = \log_2 n$ columns and is initialized as a zero matrix at the beginning of the process. Meanwhile, the row index numbering starts from zero and is increased from bottom to up, and analogously the column index starts from zero and is numbered from left to right. Therefore, the result matrix R_2 has $3 = 2+1$ rows and $5 = b = \log_2 32$ columns. In our detailed example in Fig. 4.3, the cumulative sum C_2 consists of three b-dimensional entries, i.e., c_{20}, c_{21}, and c_{22}. In the following, we show how to generate R_2 from C_2 step by step.

Step 1: Choose the lower and upper bound entries in C_2, i.e., the first entry c_{20} and the last entry c_{22}, and generate matrices for them. In specific, with c_{20} = "$0, 1, 1, 1, 2$", we can generate the lower bound entries in R_2 by updating

$$R_2\Big[c_{20}[k]\Big]\Big[k\Big] = \begin{cases} 1, & 0 \leq k \leq 4 \\ 0, & \text{otherwise.} \end{cases} \tag{4.6}$$

Similarly, with c_{22}, we generate the upper bound entries in R_2 by updating

$$R_2\Big[c_{22}[k]\Big]\Big[k\Big] = \begin{cases} 1, & 0 \leq k \leq 4 \\ 0, & \text{otherwise.} \end{cases} \tag{4.7}$$

Step 2: Continue to generate the matrix R_i by setting all entries between the lower bound and upper bound to ones. After this step, we have $b+1$ matrices, i.e., $\{R_0, \cdots, R_b\}$. Note that, our strategy to only find the lower and upper bounds and fill all entries between them as ones in each matrix R_i. That speeds the generation of

R_i, and its correctness for supporting the range query can be found in Theorem 4.4.1.

Step 3: Simplify each R_i to be a sparse matrix T_i for $0 \leq i \leq b$, which can be divided into two cases.

- Case 1: For R_0 and R_b, since R_0 and R_b will only be set to 1 when the range query contains the lower bound value 0 and the upper bound value $n-1 = 2^b - 1$; and set to 0 otherwise. Therefore, we can simplify R_0 and R_b to be 1×1 matrices T_0 and T_b. For example, in Fig. 4.2, both T_0 and T_5 are set to 0 because the given range query $[L, U] = [8, 16]$ does not include the lowest value 0 and the highest value 31.

- Case 2: For R_i $(1 \leq i \leq b-1)$, since the last columns of R_i can be generated by simply combining a single one and i zeros, as shown in Fig. 4.2 and Fig. 4.3 (highlighted in yellow). Therefore, this entire column and related entries can be discarded to increase communication efficiency. Furthermore, in each matrix R_i, we can drop $1 + 2 + 3 + \cdots + (i-1)$ entries, which are highlighted in grey, to obtain the corresponding matrix T_i.

Algorithm 6 Semi-triangular Matrices

Input: $C = \{C_i\} = \big\{\{c_j\}\big\} = \Big\{\big\{\{Element_k\}\big\}\Big\}$
Output: Vector <R[][]> # Vector of matrices
1: $R \leftarrow \emptyset$
2: **for each** $C_i \in C$ **do** # $0 \leq i \leq b$
3: $j \in \{0, C_i.\text{SIZE}()\}$ # Lower and upper bound entries
4: **for each** $Element_k \in c_j$ **do** # $0 \leq k \leq b-1$
5: $t \leftarrow Element_k$
6: $R_i[t][k] \leftarrow 1$
7: **for each** LowerBound $< R_i[\,][\,] <$ UpperBound **do**
8: $R_i[t][k] \leftarrow 1$ # $c_0 \leq t \leq c_{C_i.\text{SIZE}()} \wedge 0 \leq k \leq b-1$
9: **return** R

Theorem 4.4.1. *Our strategy employed for efficiently generating the matrices R_i, $i = 0, \cdots, b+1$, can accept the range queries in the form of $[L = 2^a, U = 2^{a'}]$, where $0 \leq a \leq a' < b = \log_2 n$.*

Proof. In order to prove the correctness of our strategy to accept the range query $[2^a, 2^{a'}]$, we first prove that our strategy can accept the range queries $[0, 2^a]$ and $[0, 2^{a'}]$.

For the range query $[0, 2^a]$, the parameter settings are $b = \log_2 n; [L, U] = [0, 2^a]$; the Hamming weight $\boldsymbol{\omega}$ is within $0 \leq \boldsymbol{\omega} \leq a$; and $0 \leq a < b$. First, after the Hamming weight classification, the lower bound $L_{S_{\boldsymbol{\omega}}}$ and upper bound $U_{S_{\boldsymbol{\omega}}}$ for each class $S_{\boldsymbol{\omega}}$, where $0 \leq \boldsymbol{\omega} \leq a$, are expressed as follows:

$$\boldsymbol{\omega} = 0 \Rightarrow \text{lower } L_{S_0} = \text{upper } U_{S_0} = 0, \quad i.e., \quad \underbrace{0 \cdots 0}_{b}$$

$$\boldsymbol{\omega} = 1 \Rightarrow \begin{cases} L_{S_1} = 1 : \underbrace{0 \cdots 0}_{b-1}1 \\ U_{S_1} = 2^a : \underbrace{0 \cdots 0}_{b-a-1}1\underbrace{0 \cdots 0}_{a} \end{cases}$$

$$\boldsymbol{\omega} = \boldsymbol{\omega} \ \Rightarrow \begin{cases} \text{lower } L_{S_{\boldsymbol{\omega}}} : \underbrace{0 \cdots 0}_{b-\boldsymbol{\omega}}\underbrace{1 \cdots 1}_{\boldsymbol{\omega}} \\ \text{upper } U_{S_{\boldsymbol{\omega}}} : \underbrace{0 \cdots 0}_{b-a}\underbrace{1 \cdots 1}_{\boldsymbol{\omega}}\underbrace{0 \cdots 0}_{a-\boldsymbol{\omega}} \end{cases} \qquad 1 < \boldsymbol{\omega} \leq a$$

Then, the corresponding cumulative sums are defined as

$$\boldsymbol{\omega} = 0 \Rightarrow \text{lower } L_{C_0} = \text{upper } U_{C_0} = 0, \quad i.e., \quad \underbrace{0 \cdots 0}_{b}$$

$$\boldsymbol{\omega} = 1 \Rightarrow \begin{cases} L_{C_1} = 1 : \underbrace{0 \cdots 0}_{b-1}1 \\ U_{C_1} = 2^a : \underbrace{0 \cdots 0}_{b-a-1}1\underbrace{\cdots 1}_{a+1} \end{cases}$$

$$\boldsymbol{\omega} = \boldsymbol{\omega} \Rightarrow \begin{cases} \text{lower } L_{C_{\boldsymbol{\omega}}} : \underbrace{0 \cdots 0}_{b-\boldsymbol{\omega}}1\underbrace{2 \cdots \boldsymbol{\omega}}_{\boldsymbol{\omega}} \\ \text{upper } U_{C_{\boldsymbol{\omega}}} : \underbrace{0 \cdots 0}_{b-a}\underbrace{1 2 \cdots \boldsymbol{\omega}}_{\boldsymbol{\omega}}\underbrace{\boldsymbol{\omega} \cdots \boldsymbol{\omega}}_{a-\boldsymbol{\omega}} \end{cases} \qquad 1 < \boldsymbol{\omega} \leq a$$

For any value $x \in [0, n-1]$, if its Hamming weight $\boldsymbol{\omega}$ is less than or equal a, then x must belong to the class $S_{\boldsymbol{\omega}}$. More accurately, it is easy to see that the value of x lies in the range $[L_{S_{\boldsymbol{\omega}}}, U_{S_{\boldsymbol{\omega}}}]$. As shown in Fig. 4.4, the cumulative sums $L_{C_{\boldsymbol{\omega}}} = \underbrace{0 \cdots 0}_{b-\boldsymbol{\omega}}\underbrace{1 2 3 \cdots \boldsymbol{\omega}}_{\boldsymbol{\omega}}, U_{C_{\boldsymbol{\omega}}} = \underbrace{0 \cdots 0}_{b-a}\underbrace{1 2 3 \cdots \boldsymbol{\omega}}_{\boldsymbol{\omega}}\underbrace{\boldsymbol{\omega} \cdots \boldsymbol{\omega}}_{a-\boldsymbol{\omega}}$ of $S_{\boldsymbol{\omega}}$'s lower and upper bounds $L_{S_{\boldsymbol{\omega}}}, U_{S_{\boldsymbol{\omega}}}$ form a shaded parallelogram area. Because all elements in $S_{\boldsymbol{\omega}}$ have the same weight ($\boldsymbol{\omega}$), their cumulative sums will finally reach the same point $R_{\boldsymbol{\omega}}[\boldsymbol{\omega}][b] = 1$ in the matrix $R_{\boldsymbol{\omega}}$. Therefore, if x really belongs to $[L_{S_{\boldsymbol{\omega}}}, U_{S_{\boldsymbol{\omega}}}]$, its cumulative sum will be within the shaded parallelogram area in Fig. 4.4. As a result, our strategy of only finding the lower and upper bounds and filling all entries between them as ones in each matrix $R_{\boldsymbol{\omega}}$ can accept the range query with $[L, U] = [0, 2^a]$. Obviously, our strategy can also accept the range query $[0, 2^{a'}]$. Next, we prove that our strategy can accept the range query $[L = 2^a, U = 2^{a'}]$.

For any value $x \in [0, n-1]$, if its Hamming weight is $\boldsymbol{\omega}$ and $\boldsymbol{\omega} \leq a \leq a'$, then x must belong to the class $S_{\boldsymbol{\omega}}$. Based on the above discussion, the range $[0, 2^{a'}]$ can generate a parallelogram with the length $(a' - \boldsymbol{\omega})$ times the height $\boldsymbol{\omega}$, and the range $[0, 2^a]$ can generate a parallelogram with the length $(a - \boldsymbol{\omega})$ times the same height $\boldsymbol{\omega}$ in the matrix $R_{\boldsymbol{\omega}}$, as shown in Fig. 4.5. Hence, if x really belongs to the range $[L = 2^a, U =$

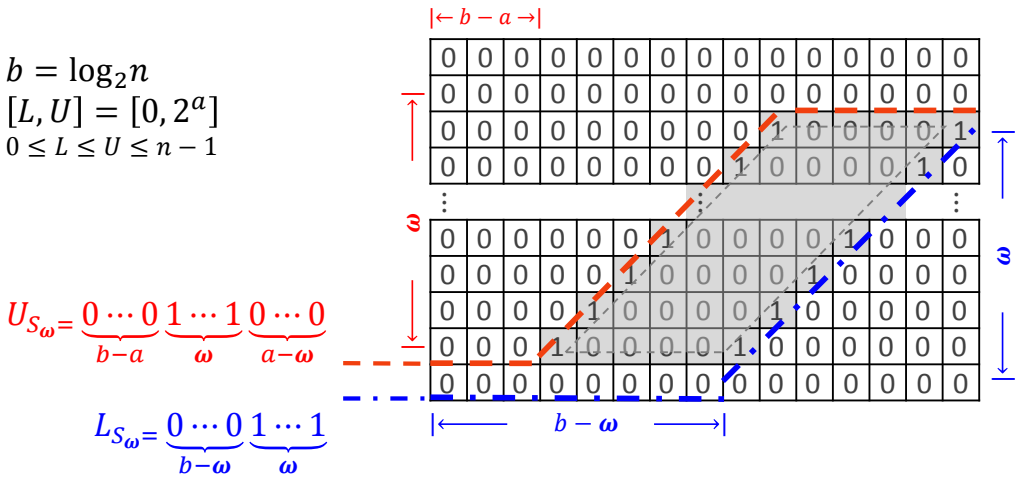


Figure 4.4: Parallelogram area enclosed between the lower and upper bound values for the Hamming weight class S_ω range query $[L, U] = [0, 2^a]$

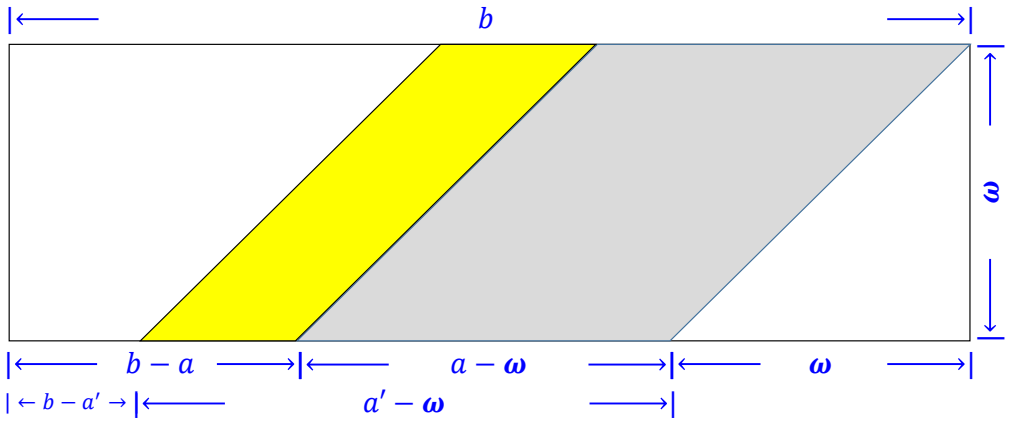


Figure 4.5: Yellow-shaded parallelogram area enclosed between the lower and upper bound values for the Hamming weight class S_ω in range query $[L, U] = [2^a, 2^{a'}]$

$2^{a'}]$, its cumulative sum will be within the yellow-shaded parallelogram area in the matrix R_ω. As a result, for each matrix R_ω, where $\omega \le a \le a'$, we can first find the lower bound $L_{S_\omega} : \underbrace{0\cdots0}_{b-a}\underbrace{1\cdots1}_{\omega}\underbrace{0\cdots0}_{a-\omega}$, the upper bound $U_{S_\omega} : \underbrace{0\cdots0}_{b-a'}\underbrace{1\cdots1}_{\omega}\underbrace{0\cdots0}_{a'-\omega}$, and fill all entries between them as ones can accept the range query $[L, U] = [2^a, 2^{a'}]$.

For any value $x \in [0, n-1]$, if its Hamming weight is between a and a' ($a < \omega \le a'$), then x must belong to the class S_ω. Because $\omega > a$, the range $[0, 2^a]$ does not contribute any element in class S_ω. Therefore, all elements in S_ω are generated from the range $[0, 2^{a'}]$. As a result, for each matrix R_ω, where $a < \omega \le a'$, we can first find

the lower bound L_{S_ω} : $\underbrace{0\cdots01\cdots1}_{b-\omega\quad\omega}$, the upper bound U_{S_ω} : $\underbrace{0\cdots01\cdots10\cdots0}_{b-a'\quad\omega\quad a'-\omega}$, and fill all entries between them as ones can accept the range query $[L, U] = [2^a, 2^{a'}]$. By summarizing all of the above discussions, we complete the proof of Theorem 4.4.1. □

4.4.2 Communication Overhead of the Decomposition Technique

Since the novelty of our $O(\log^3 n)$ communication-efficient range query derives its efficiency from our novel decomposition process, here we will briefly review the decomposition steps and prove its efficiency. Our decomposition technique will convert the original range query $[L, U]$ into corresponding matrices, namely semi-triangular matrices R_i's which ultimately result in sparse matrices T_i's. In the following, we will determine the exact number of ciphertexts to form the distinct semi-triangular and sparse matrices. First, both T_0 and $T_{b=\log n}$ are 1×1 matrices, so the communication cost (denoted as $\mathbb{C}$) is $\mathbb{C} = 2$. Second, the remaining R_i's, $1 \leq i \leq b-1$ can be organized by $b-1$ matrices in which each one has b columns. Clearly, the last column, cells shaded in yellow color, can be discarded and since each matrix R_i and the corresponding T_i contain $i+1$ rows, therefore the total number of required ciphertexts in T_i is $(b-1) * (i+1)$. Up to this point, the communication cost is $\mathbb{C} = 2 + \sum_{i=1}^{b-1}(b-1)(i+1)$. Finally, the unused area of each T_i should be eliminated with a considerable reduction in the number of transmitted ciphertexts. Specifically, $1+2+\cdots+(i-1) = \frac{i(i-1)}{2}$ reflects the appropriate number of shaded cells in R_i that can be eliminated in this step to form each T_i. As a result, the proposed algorithm's communication cost $\mathbb{C}$ is

$$\begin{aligned}\mathbb{C} &= 2 + \sum_{i=1}^{b-1}(b-1)(i+1) - \sum_{i=1}^{b-1}\frac{i(i-1)}{2} \\ &= 2 + \frac{(b-1)^2(b+2)}{2} - \frac{(b-1)b(b-2)}{6} \\ &= 2 + \frac{2b^3+3b^2-11b}{6} \in O(\log^3 n)\end{aligned}$$

4.4.3 Description of Our Proposed Scheme

We will now give a detailed description of our communication-efficient privacy-preserving range query scheme in fog-based IoT. It consists of five phases: 1) querying user key generation, 2) range query generation at querying user, 3) query response at IoT devices, 4) response aggregation at the fog node, and 5) response recovery at querying user.

4.4.3.1 Querying User Key Generation

For simplicity's sake, we consider $n = 2^b$ to represent n as a b-bit binary value. Given the input parameters (k_0, k_1, k_2), the querying user generates the secret key $SK = (p, q, \mathcal{L})$ and the public parameter PP based on our previously described SHE scheme. Then, the querying user keeps the secret key SK and sends the public parameter PP to the fog node and IoT devices $\mathbf{I} = \{I_1, I_2, \cdots, I_N\}$. In order to achieve $O(\log^3 n)$ communication cost, the following two conditions should be satisfied:

- The message space depends on k_1 and it should be set at least to $\lceil \log_2 N \rceil$ to successfully recover the query response $\text{COUNT}(\mathbf{I}') = |\mathbf{I}'|$, where N is the number of IoT devices.

- The SHE scheme should accept at least b homomorphic multiplication operations, therefore k_0 should be set equal to $2(b+1)k_2$ in order to successfully

evaluate the b-depth multiplicative circuit.

4.4.3.2 Range Query Generation at Querying User

When the querying user intends to request a range query $[L, U]$, where $0 \leq L \leq U \leq n-1 = 2^b - 1$ and $L = 2^a, U = 2^{a'}$, he/she can generate the query request in the following steps.

Step 1: Transform the range $[L, U]$ to be $b+1$ matrices $\{T_0, T_1, \cdots, T_b\}$ by following the decomposition technique in Section 4.4.1.

Step 2: Encrypt each T_i for $0 \leq i \leq b$. In specific, in the matrix T_i, the value zero and one is encrypted to be $Enc(0)$ and $Enc(1)$ with the encryption method of our proposed SHE. Then, the querying user forwards the encrypted sparse matrices $\{Enc(T_0), Enc(T_1), \cdots, Enc(T_b)\}$ to the IoT devices via fog node. At the same time, besides the encrypted T_i's, an extra $Enc(0)$ is also generated and sent to each IoT device, which can provide the self-blindness functionality for our scheme. For example, the IoT device can multiply the additional $Enc(0)$ to a random integer value, and then the result is added to encrypted prepared response to re-randomize the response for protecting against the curious fog node.

4.4.3.3 Query Response at IoT Devices

Upon receiving encrypted sparse matrices $\{Enc(T_0), Enc(T_1), \cdots, Enc(T_b)\}$, each IoT device I_i with prepared data x_i will perform the following steps to report its encrypted response. Note that, though the employed SHE is a symmetric encryption algorithm, when each IoT device reports its encrypted response, it does not need to know the secret key SK. Instead, each IoT device can directly perform over the ciphertexts. We will illustrate each step by using two different examples, where x_i is within the range and out of the range. Recall the example in Fig. 4.2, where the user range is bounded within $[8, 16], 0 \leq L \leq U \leq n-1 = 31$ and consider two different

x_is, e.g., 7 is within, and 10 is out of the range.

Step 1: IoT device I_i converts its prepared data x_i into binary form, e.g., $7 = (00111)$ and $10 = (01010)$.

Step 2: Compute the Hamming weight of the binary value x_i, i.e., $\boldsymbol{\omega} = \textsc{HammingWeight}(x_i)$, which is calculated to indicate the index of sparse matrix T_i that should be used to calculate the encrypted result, e.g., $HW\Big(7 = (00111)\Big) = 3$ and $HW\Big(10 = (01010)\Big) = 2$, therefore IoT device will investigate T_3 and T_2, respectively for $x_i = 7$ and $x_i = 10$.

Step 3: The IoT device generates the cumulative sum value with respect to x_i, i.e., $\beta_{[0..b-1]} = \textsc{CummulativeSum}(x_i)$. For example, the cumulative sum for $x_i = 7$ is 00123 and in case of $x_i = 10$ is 01122.

Step 4: The IoT device I_i calculates the following multiplicative expression as the encrypted response.

$$Response_{(I_i)} = \prod_{k=0}^{b-1} T_\alpha[\beta(k)][k]$$

For example, the encrypted response in case of $x_i = 7 = (00111)_2$ is

$$\begin{aligned}
Response_{(I_i)} &= \prod_{k=0}^{4} T_\alpha[\beta(k)][k] \\
&= T_3[\beta(0)][0] \times T_3[\beta(1)][1] \times \cdots T_3[\beta(4)][4] \\
&= T_3[0][0] \times T_3[0][1] \times T_3[1][2] \times T_3[2][3] \times T_3[3][4] \\
&= Enc(1) \times Enc(0) \times Enc(1) \times Enc(1) \times Enc(1) \\
&= Enc(0) \therefore 7 \notin [8, 16]
\end{aligned}$$

While for $x_i = 10 = (01010)_2$, the encrypted response is

$$\begin{aligned} Response_{(I_i)} &= \prod_{k=0}^{4} T_\alpha[\beta(k)][k] \\ &= T_2[\beta(0)][0] \times T_2[\beta(1)][1] \times \cdots T_2[\beta(4)][4] \\ &= T_2[0][0] \times T_2[1][1] \times T_2[1][2] \times T_2[2][3] \times T_2[2][4] \\ &= Enc(1) \times Enc(1) \times Enc(1) \times Enc(1) \times Enc(1) \\ &= Enc(1) \therefore 10 \in [8, 16] \end{aligned}$$

Step 5: To protect $Response_{(I_i)}$ from a curious fog node, the IoT device I_i self-blinds $Response_{(I_i)}$ with the extra prepared $Enc(0)$ as

$$Response_{(I_i)} = Response_{(I_i)} + \Big(Enc(0) \times \mathsf{r}\Big)$$

where $\mathsf{r} \in \mathcal{M}$. Finally, I_i sends the self-blinded encrypted response $Response_{(I_i)}$ to the fog node.

4.4.3.4 Response Aggregation at Fog Node

After receiving all self-blinded responses from IoT devices, the fog node aggregates the intermediate encrypted responses to generate the final query result as

$$Count = \sum_{I_i \in \mathbf{I}} Response_{(I_i)}$$

Then, the fog device forwards $Count$ to the user.

4.4.3.5 Response Recovery at Querying User

On receiving the encrypted query result $Count$, the user uses secret key SK to recover the range query response as

$$Count \xrightarrow{Dec} \text{Count}(\mathbf{I}') = |\mathbf{I}'| = \sum_{I_i \in \mathbf{I}} Response_{(I_i)}$$

The correctness of the result is:

$$\begin{aligned} Count &= \sum_{I_i \in \mathbf{I}} Response_{(I_i)} = \sum_{I_i \in \mathbf{I}} \left(\prod_{k=0}^{b-1} T_\alpha[\beta(k)][k] \right) \\ &= \sum_{I_i \in \mathbf{I}'} Enc(1) + \sum_{I_i \notin \mathbf{I}'} Enc(0) = Enc(|\mathbf{I}'|) \end{aligned}$$

4.5 Security Analysis

In this section, we analyze the security of our proposed privacy-preserving range query scheme. Particularly, we focus on the privacy properties, i.e., range query $[L, U]$ and the subset $\mathbf{I}'$ should be privacy-preserving.

• *The query range* $[L, U]$ *is privacy-preserving in the proposed scheme.* As described in Section 4.4.3, in order to achieve communication efficiency, the query range $[L, U]$ will be transformed into $b + 1$ sparse matrices $\{T_0, T_1, \cdots, T_{b=\log_2 n}\}$. At the same time, each value in matrix T_i will be encrypted by our SHE cryptosystem, which shown to be secure against the known-plaintext attack in Theorem 4.3.2. Then, the security of our SHE cryptosystem guarantees that without knowing the secret key $SK = (p, q, \mathcal{L})$ the adversary has no idea on the matrix T_i for $0 \leq i \leq b$. Since both the fog device and IoT devices cannot access to the secret key SK, they have no idea on the matrix T_i $(0 \leq i \leq b)$, i.e., they cannot distinguish whether a ciphertext in encrypted T_i is encrypted from zero or one. Furthermore, without any plaintext information about each T_i, the fog node and IoT devices definitely have no idea on the query range $[L, U]$. Therefore, the query range $[L, U]$ in the proposed

scheme can be kept secret from both the fog node and IoT devices, as a result it is privacy-preserving.

• *The subset* $\boldsymbol{I'}$ *is also privacy-preserving in the proposed scheme.* In the proposed scheme, the subset $\mathbf{I}'$ denotes a set of IoT devices whose data are within the query range $[L, U]$, i.e., $\mathbf{I}' = \{I_i | I_i \in \mathbf{I} \wedge x_i \in [L, U]\}$. As described in our security model, $\mathbf{I}'$ should be kept secret from the querying user, fog device, and IoT devices. For each IoT device I_i, if it attempts to obtain the information about $\mathbf{I}'$, it needs to determine whether itself is in $\mathbf{I}'$ or not and whether other IoT devices are in $\mathbf{I}'$ or not. For I_i, it uses its prepared data x_i to compute the encrypted $Response_{(I_i)}$, but has no idea on $Response_{(I_i)}$ is $Enc(0)$ or $Enc(1)$. In other words, I_i does not know whether w_i is in the query range $[L, U]$ or if I_i is in $\mathbf{I}'$. For the other IoT devices, since I_i cannot access the plaintext data prepared by them and the plaintext of query range, it also has no idea on whether other IoT devices are in $\mathbf{I}'$ or not. Thus, the subset $\mathbf{I}'$ can be kept secret from the IoT devices.

For the fog node, it receives the encrypted query range $[L, U]$ from the user and the encrypted response $Response_{(I_i)}$ from each IoT device I_i. It is likely to use them to deduce the information of the subset $\mathbf{I}'$. However, on the one hand, the query range $[L, U]$ is transformed into matrices $\{T_0, T_1, \cdots, T_b\}$ and are encrypted by our SHE cryptosystem. As analyzed above, our SHE cryptosystem guarantees that the fog node has no idea of the plaintext of the query range. On the other hand, each IoT device's encrypted response $Response_{(I_i)}$ is also encrypted by our SHE cryptosystem. At the same time, the $Response_{(I_i)}$ is self-blinding with the factor $\Big(Enc(0) \times \mathsf{r}\Big)$. Note that, if there is no self-blinding factor $\Big(Enc(0) \times \mathsf{r}\Big)$, it is possible for the fog node to identify the specific IoT device's response. Specifically, it can iterate over all possible prepared values x_i $(0 \leq x_i \leq n-1)$ by generating Hamming weight and cumulative sum values as well as multiplying the corresponding sparse matrix entries to check whether $Response_{(I_i)} \stackrel{?}{=} \prod_{k=0}^{b-1} T_\alpha[\beta(k)][k]$. Although

it requires a lot of computations, it is feasible, especially when n is small value. Luckily, $Response_{(I_i)} = Response_{(I_i)} + \Big(Enc(0) \times \mathsf{r}\Big)$ includes the random factor $\Big(Enc(0) \times \mathsf{r}\Big)$, which makes it impossible for the fog node to launch a brute force attack on IoT responses. In this case, it is almost impossible for the fog node to obtain the information about the subset $\mathbf{I}'$. Thus, the subset $\mathbf{I}'$ can be kept secret from the fog node.

For the querying user, he/she receives the encrypted query response from the fog node and recovers the plaintext of the query result by decryption. As described in Section 4.4.3, the query response is an aggregated result, i.e., $|\mathbf{I}'|$. Thus, the querying user only knows the number of IoT devices whose data are within the query range, but has no idea which specific IoT device has a data within the query range. Thus, the subset $\mathbf{I}'$ can be kept secret from the querying user under our considered security model.

4.6 Performance Evaluation

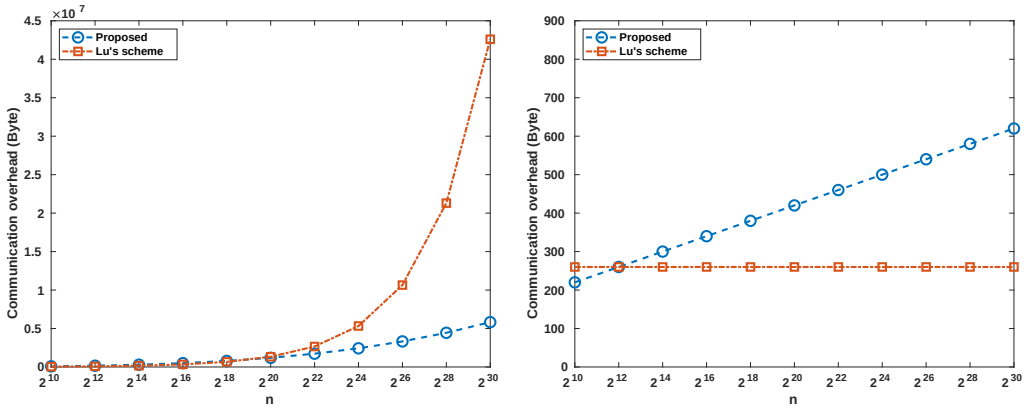
In this section, we evaluate the performance of our proposed scheme with respect to the communication overhead and computational cost. Specifically, we compare our proposed scheme with a recently proposed privacy-preserving range query scheme, i.e. Lu's scheme [85]. Lu's privacy-preserving range query scheme with $O(\sqrt{n})$ communication complexity is performed based on BGN homomorphic encryption [21]. In our performance evaluation, we implemented both our scheme and Lu's scheme in Java (JDK 1.8 update 222) on a machine with Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz, 8 GB main memory, and Linux (Ubuntu 16.04) operating system. The detailed parameter settings for both schemes are listed in Table 4.2. We repeated each experiment 10 times and reported the average results.

Table 4.2: The parameter settings

Shared param.	Value
N	The number of IoT devices, $N = 1000$
n	The upper bound of $[0, n-1]$, $n = \{2^{10}, 2^{12}, 2^{14}, 2^{16}, 2^{18}, 2^{20}, 2^{22}, 2^{24}, 2^{26}, 2^{28}, 2^{30}\}$
$[L, U]$	The range query lower and upper bound, $0 \leq L \leq U \leq n-1$
w_i	IoT device D_i's prepared data, randomly picked from $[0, n-1]$ where $0 \leq w_i \leq n-1$

Proposed scheme based on SHE	
Parameter	Value
b	$b = \log_2(n)$, $b = \{10, 12, 14, \cdots, 30\}$
k_0, k_1, k_2	$k_2 = 40$, $k_1 = \log_2 N$, $k_0 = 2(b+1)k_2$
p, q ,$\mathcal{L}$, $\mathcal{N}$	$\lvert p\rvert = \lvert q\rvert = k_0$, $\lvert\mathcal{L}\rvert = k_2$, $\mathcal{N} = pq$
$\mathcal{M}$	The query result $\mathcal{M} = N \in \{0,1\}^{k_1}$

Lu's scheme based on BGN Cryptosystem	
Parameter	Value
b	$b = \sqrt{n}$, $b = \{2^5, 2^6, 2^7, \cdots, 2^{15}\}$
κ	$\kappa = 512$
p, q, $\mathcal{N}$	$\lvert p\rvert = \lvert q\rvert = \kappa = 512$, $\mathcal{N} = pq$
Δ	The query result's upper bound, $\Delta = N$



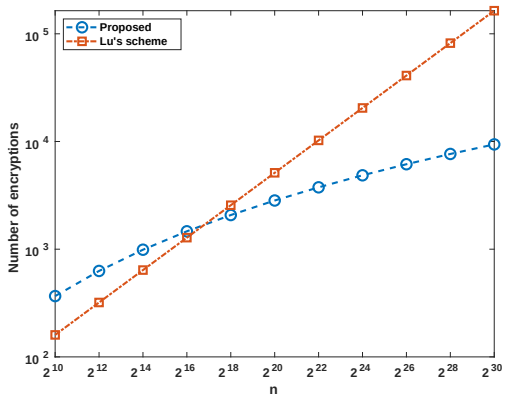
(a) Request from user to fog node

(b) Response from fog node to user

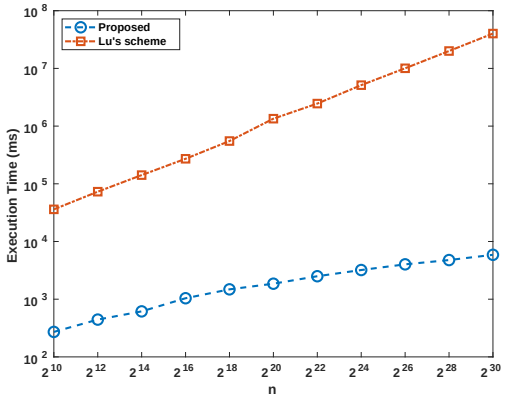
Figure 4.6: The communication overhead comparison between our proposed scheme and Lu's scheme with respect to n varying from 2^{10} to 2^{30}

4.6.1 Communication Overhead

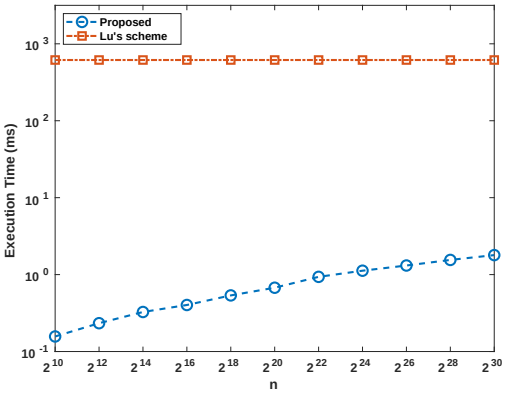
In this subsection, we evaluate the communication overhead of both the query request and the query response. First, Fig. 4.6(a) plots the communication overhead of transferring an encrypted query request from a user to fog node in both schemes with n varying from 2^{10} to 2^{30}. From the figure, we see the communication overhead of Lu's scheme is growing exponentially, while ours remains very low and highly efficient. This is because Lu's scheme has to upload $O(\sqrt{n})$ encrypted data from the user to the fog node, while ours only uploads $O(\log^3 n)$ encrypted data. Second, Fig. 4.6(b) shows the communication overhead of a query response from the fog node to the user in both schemes with respect to n. One can see that both schemes



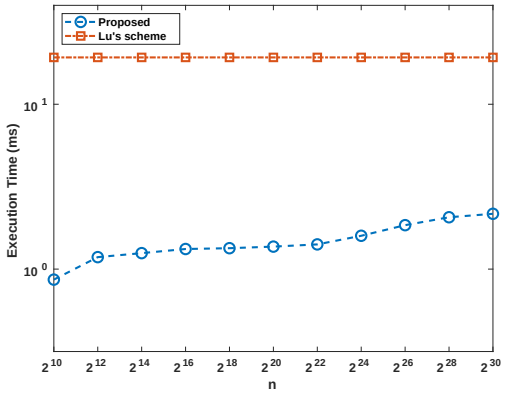
(a) The number of encryptions in the decomposition phase



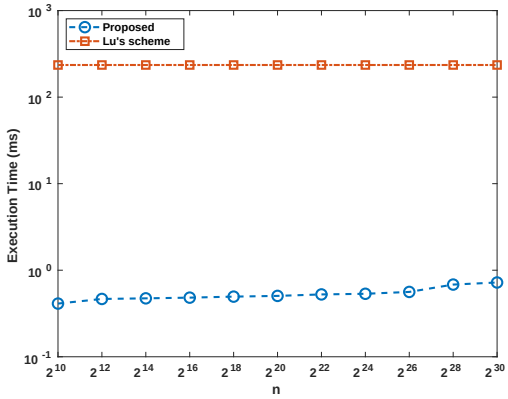
(b) Query generation at the user side



(c) Query response at the IoT device



(d) Response aggregation at the fog node



(e) Response recovery at the user side

Figure 4.7: The computational cost comparison between our scheme and Lu's scheme with respect to n varying from 2^{10} to 2^{30}

are communication efficient. Although the response cost in our proposed scheme increases with n, it is acceptable and is as low as almost 600 bytes for $n = 2^{30}$.

4.6.2 Computational Cost

The computational cost of both schemes is influenced by the number of encryptions performed by the user during the range query formulation and decomposition phase. We first compare the number of encryption operations in both schemes with respect to n as shown in Fig. 4.7(a). From this figure, we can see that the number of encryptions in Lu's scheme is rapidly increasing with n. This is because of the difference in the underlying decomposition technique. Lu's proposed decomposition technique will lead to $O(\sqrt{n})$ communication cost, whereas our scheme results in $O(\log^3 n)$ communication cost.

We now evaluate the computational cost of both schemes at each step of range query processing with respect to n as shown in Fig. 4.7. Specifically, Fig. 4.7(b) shows the query generation execution time, which indicates that the computational cost of Lu's scheme grows exponentially with n, while ours grows much more slowly due to ours having fewer encryptions. Fig. 4.7(c) illustrates the query response time of both schemes for different problem sizes. Although the processing time in Lu's scheme is constant, it is still considerably higher than ours (only one millisecond for $n = 2^{30}$). This is mainly because of the time-consuming pairing operations and costly homomorphic calculations in BGN compared with fast and efficient multiplicative homomorphic operations in our SHE scheme.

Fig. 4.7(d) depicts the response aggregation time at fog node. Merging the IoT devices' responses at fog node is mainly affected by the number of IoT devices, i.e. N. Therefore, as it is apparent from the figure, the aggregation time in both schemes is independent of problem size, even if there are small increments in our proposed scheme, varying with n. Our scheme has insignificant changes with respect to n due to gradually increasing ciphertext lengths, while in Lu's scheme, the ciphertext length only depends on κ value and is consequently independent of n.

Finally, Fig. 4.7(e) presents the response recovery time, which measures how long

time the user spent decrypting the ciphertext upon receiving the encrypted result from fog node. Obviously, they are still efficient and independent of problem size n. However, decryption in our proposed scheme is faster than Lu's. Because Lu's scheme performs in BGN and it highly depends on the message space Δ, it can only operate effectively with a relatively smaller Δ.

4.7 Related Work

Recently, some related works have emerged on privacy-preserving range queries in the fog-based IoT [85], which are closely related to our proposed query scheme. As discussed in Section 4.1, in order to achieve the desired security properties for a privacy-preserving range query, the naive solution has an $O(n)$ communication cost for the range query. For a toy example with $n = 10$, the naive solution is efficient enough. However, when n becomes larger, the efficiency, especially the communication efficiency rapidly deteriorates. To improve this, Lu [85] proposed a new privacy-preserving range query scheme was built upon BGN homomorphic encryption together with a novel range query expression, decomposition, and composition technique. It had an $O(\sqrt{n})$ communication efficiency. Therefore, when n becomes larger, Lu's scheme is much more efficient than the naive solution in terms of communication overhead. However, when n becomes even larger, e.g., $n = 2^{30}$, the $O(\sqrt{n})$ communication cost seems too costly. In addition, because Lu's scheme is based on the BGN with time-consuming bilinear pairing operations in public key settings, the computational cost is also expensive. Aiming to improve the communication efficiency, in this chapter, we proposed a more efficient privacy-preserving range query scheme in the fog-based IoT. Using a novel range query decomposition technique and an efficient symmetric homomorphic scheme (SHE), our proposed scheme achieves $O(\log^3 n)$ communication-efficiency.

When the privacy-preserving range query on "*the number of IoT devices whose data x_i is within the range [L, U]*" is considered as a special privacy-preserving data aggregation scheme in fog-based IoT, there are other related works close to our proposed scheme. In [86], Lu et al. addressed heterogeneous data aggregation in real IoT applications by proposing a light-weight privacy-preserving data aggregation (LPDA) for fog-enabled setting. The proposed LPDA is characterized by applying Paillier cryptosystem, Chinese Remainder Theorem, and a one-way hash chain function to aggregate hybrid IoT devices' data and to early filter the injected false data at the network edge. In [60], Huang et al. studied the fog-assisted selective aggregation operation. Specifically, they constructed a new threat model to formalize the non-collusive and collusive attacks of compromised fog nodes, and their proposed privacy-preserving and reliable selective multi-source aggregation scheme, which is comprised of BCP cryptosystem [26], randomized message-lock encryption, homomorphic proxy-authenticators, and multi-dimensional aggregation, can well tackle the data privacy and reliability challenges. Most recently, authors in [90] presented a privacy-preserving subset aggregation scheme in fog-enhanced IoT scenarios, which enables a query user to gain the sum of the prepared data from a subset of IoT devices. To identify the subset, the inner product similarity of the normalized vectors in the query user side and each IoT device is securely computed. Only when the inner product is greater than the user's specified threshold will an IoT device's data be privately aggregated to form the final response.

4.8 Concluding Remarks

In this chapter, we have proposed a communication-efficient privacy-preserving range query scheme for fog-based IoT. For the range queries of the form $[L = 2^a, U = 2^{a'}]$, the proposed scheme is characterized by designing a novel range decomposition tech-

nique to reduce the communication overhead to be $O(\log^3 n)$, and devising an efficient homomorphic encryption scheme to preserve the data privacy for both the range query and individual IoT device's data. To the best of our knowledge, the communication overhead of our proposed scheme is much lower than that of the most communication efficient privacy-preserving range query scheme, i.e., Lu's scheme, whose communication overhead is $O(\sqrt{n})$. Detailed security analysis shows that our scheme is indeed privacy-preserving under our defined security model. An extensive performance evaluation validated the efficiency of our scheme in terms of communication overhead and computational cost. In our next chapter, we plan to expand this study by: 1) investigating a new approach to accept flexible form of range queries, 2) launching more efficient and lower cost decompostion/compostion technique with $O(\log^2 n)$ communication cost.

Chapter 5

Using Reduced Paths to Achieve Efficient Privacy-Preserving Range Query in Fog-Based IoT

5.1 Introduction

Recent advancements in Internet of Things (IoT) and wireless communication technologies have enabled us to enjoy various smart things around our daily lives, including smart vehicles [67, 69, 76], smart homes [34, 77, 147], smart grids [15, 14, 16, 122], and smart healthcare [116]. All of these are highly dependent upon the massive data generated by IoT. However, due to the limited built-in storage and computing capability of IoT devices, the data usually need to be transferred from IoT devices to cloud servers for storage and processing, which will inevitably encounter bandwidth problems and incur response latency [145]. In order to address these bandwidth and latency issues, the concept of fog computing was proposed by Cisco in 2012 [86]. Essentially, the idea is to deploy fog devices equipped with certain storage and computing capabilities at the network edge, which can directly solve problems or pre-

process parts of a problem close to the IoT domain. Thus, it can provide low latency responses and improve communication efficiency by only forwarding pre-processed data to cloud server. Over the past years, IoT integrated with fog computing has attracted considerable attention. However, since fog devices are deployed at the network edge and might not be fully trusted, fog-based computing still faces some security and privacy issues [17, 49]. In this chapter, we aim to insure privacy while also being efficient when performing privacy-preserving range aggregate queries in fog-based IoT scenarios. To clearly illustrate the research problem in fog-based IoT, let us consider the following scenario in a smart grid [16, 14]. For the better electricity planning, the manager (i.e., a query user) of a smart grid may query the number of houses whose individual electricity consumption is within a certain range in a certain area for planning purposes. If neither security nor privacy are of concerns, this query can be easily completed. The query user sends the range to the fog devices and the fog devices can directly return the query responses through comparing the range with the values from the IoT devices (i.e., smart meters). However, when the privacy is taken into consideration, the values from the meters may reveal the resident's privacy, such as whether he/she lived in the house recently. Moreover, the query and the corresponding result may also reveal the query user's privacy. For example, his/her selling strategy could be deduced from the query and result. For reasons such as these, several solutions have been proposed [85, 89]. However, all of them suffer from one or more of the following problems. i) *Privacy level:* the fog devices in many solutions [111, 78, 75] can deduce the query result from the communications between the query user and IoT devices; ii) *High communication overhead:* the communication complexity in a straightforward solution [85] is $O(n)$, where n is the biggest value that the smart meter can return in a certain period of time. The current best result with semi-triangular sparse matrices (STSM) [89] is $O(\log^3 n)$ communication efficiency, which seems still high for large n [89]; and iii)

Limited query types: most of the current solutions cannot support non-continuous range queries directly, and the only work-around is to run the original solution repeatedly. However, the resulting solution would reveal some private information to the fog device, e.g., how many ranges are in the query. Aiming to solve the above issues, in this chapter, we employ a reduced path technique to implement a new efficient and privacy-preserving range query scheme for fog-based IoT. Specifically, the contributions of this chapter are threefold:

• First, we present a new privacy-preserving range query solution for the fog-based IoT. The notable property of our proposal is that the communication complexity is only $O(\log^2 n)$. To the best of our knowledge, it is the best result for privacy-preserving range queries in the fog-based IoT setting.

• Second, our proposal is the first to support continuous and non-continuous queries and aggregating queries at the same time. The decomposition technique addresses query format of recent studies [89, 85]. Also, [89] suffers from a limitation in the query form in which the lower and upper bound values must be of the form of two to the power of integer value, i.e., $[L, U] = [2^a, 2^{a'}]$. The critical component, which may be of independent interest, is the use of encrypted inverted reduced paths. Range queries are decoded as reduced paths in a perfect binary tree, where all the nodes are encrypted with the symmetric homomorphic encryption (SHE) scheme proposed in our previous study [89].

• Third, the detailed security analysis shows that our solution is indeed privacy-preserving scheme. Furthermore, extensive experiments also demonstrate that our proposal outperforms state-of-the-art solutions in terms of low communication complexity [85, 89]. The remainder of the chapter is structured as follows. We first introduce our system model, security model, and design goal in Section 5.2. Then, we recall our preliminaries in Section 5.3. After that, we present our proposed scheme in Section 5.4, followed by security analysis and performance evaluation in Section

5.5 and Section 5.6, respectively. Some related works are discussed in Section 5.7, and, finally, conclusions are summarized in Section 5.8.

5.2 Models and Design Goal

In this section, we explain our system model, security model, and set out the design goal for our communication-efficient privacy-preserving range query. For a clear description, some used symbols are first listed in Table 5.1.

Table 5.1: The list of symbols used in the proposed tree-based scheme

Symbol	Description
N	The number of the IoT devices.
$\mathbf{I}, I_i$	The i-th IoT device in set $\mathbf{I} = \{I_1, I_2, \cdots, I_N\}$, $1 \leq i \leq N$.
n	The maximum possible value that can be sensed by IoT devices.
w_i	Sensed and prepared data in I_i ranging from 0 to $n-1$; $0 \leq w_i \leq n-1$.
L, U	Lower and upper bound in a range query, $0 \leq L \leq U \leq n-1$.
$\mathbf{I}'$	The subset of $\mathbf{I}$ where I_i's data $w_i \in [L, U]$.
PP	Public parameter in SHE scheme, i.e., $PP = (k_0, k_1, k_2, \mathcal{N})$.
SK	Secure secret key in SHE scheme; $SK = (p, q, \mathcal{L})$.
$E(), D()$	Encryption and Decryption functions.
T	The corresponding perfect binary tree with the whole range $[0, n-1]$.
S	Paths from root to leaves in T where leaf node $t \in [L, U]$.
R	Reduced path strings, extracted from S.
V	Inverted path strings, by inverting 0s to 1s and vice versa in R.
Q	Ciphertext equivalent of V, by separately encrypting 0s and 1s in V.
F	Final encrypted range query including real paths, dummy paths, and filtering column.

5.2.1 System Model

In our system model for privacy-preserving range queries in fog-based IoT, as shown in Fig. 5.1, there are three kinds of entities, namely, a query user at the user layer, a fog node at the fog layer, and a set of IoT end-devices $\mathbf{I} = \{I_1, I_2, \cdots, I_N\}$ at the end-device layer. The detailed descriptions of these entities are as follows.

• Query user: In our system model, a query user formulates and submits secure range queries to the IoT end-devices via the fog node, and also obtains the corresponding encrypted responses via the fog node. Particularly, as indicated in Fig. 5.1, the range

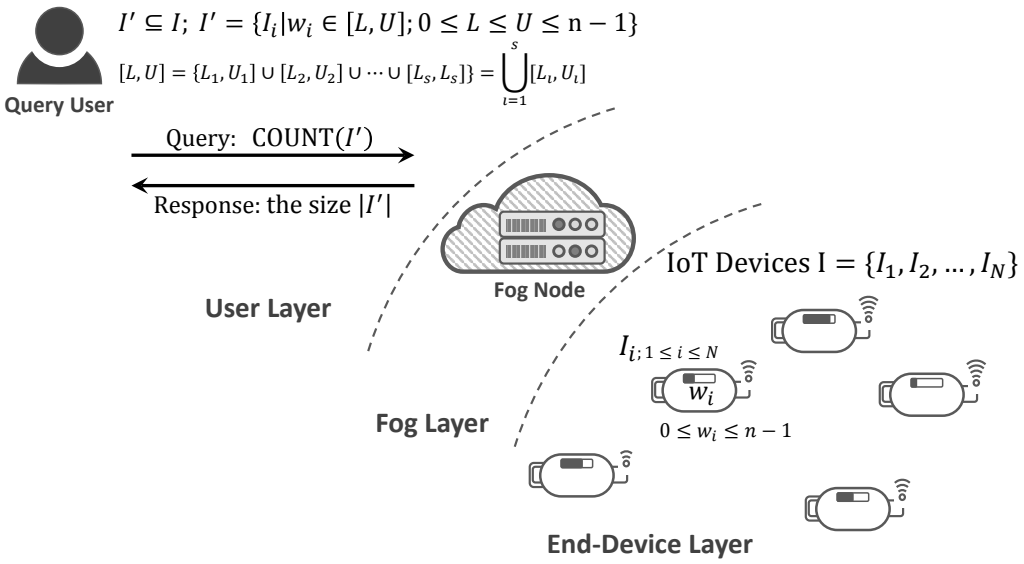


Figure 5.1: System model under consideration

count query, i.e., COUNT($\mathbf{I}'$), can be submitted to retrieve the number of IoT end-devices in subset $\mathbf{I}' \subseteq \mathbf{I}$. Formally, we consider $\mathbf{I}'$ as a subset of $\mathbf{I} = \{I_1, I_2, \cdots, I_N\}$, where the prepared data w_i at each IoT end-device $I_i \in \mathbf{I}'$ is within the range $[L, U]$, i.e.,

$$\mathbf{I}' = \{I_i | I_i \in \mathbf{I} \ \wedge w_i \in [L, U]; 0 \leq L \leq U \leq n - 1\} \tag{5.1}$$

Upon receiving the above count-query, the fog node returns the response as follows:

$$\mathrm{COUNT}(\mathbf{I}') = \text{the size}|\mathbf{I}'| \tag{5.2}$$

Note that, the range $[L, U]$ in our proposed system model may be continuous or non-continuous, i.e.,

$$[L, U] = [L_1, U_1] \cup [L_2, U_2] \cup \cdots \cup [L_s, U_s] = \bigcup_{\iota=1}^{s} [L_\iota, U_\iota] \tag{5.3}$$

For example, our proposed scheme can handle simple continuous range queries, e.g., $[L, U] = [5, 12]$, and complex non-continuous ones, e.g., $[L, U] = [0, 7] \cup [16, 19] \cup [24, 24] \cup [28, 29]$, as depicted in Fig. 5.2 and Fig. 5.3, respectively.

• Fog node: A fog node is deployed at the network edge, i.e., the fog layer, which processes data prepared by IoT end-devices and relays the aggregated results to the query user. In this model, an encrypted query from the query user is forwarded to IoT end-devices via the fog node. The IoT end-devices, then, encrypt their responses and report them to the fog node for aggregation. Finally, the fog node returns the aggregated results to the query user.

• IoT end-devices $\mathbf{I} = \{I_1, I_2, \cdots, I_N\}$: A large population of IoT end-devices ($\mathbf{I}$) of size N are spread across the end-device layer and related to the specific IoT ecosystem. After collecting the raw data w_i from sensing component, each end-device I_i can provide data processing functionality before delivering the data to the fog node. More precisely, different processing tasks such as validating, converting, packaging, and even encrypting will enable IoT end-device I_i to successfully transmit highly accurate sensitive data. For example, suppose a sensed high-precision floating-point value $w_i = 0.195874$ is to be sent. It can be modified into an arbitrary precision integer by applying simple scaling, truncating or rounding functions, e.g., $truncate(w_i = 0.195874 * 10000 = 1958.74) = 1958$. Values are then encrypted and sent to the fog node. For simplicity but without losing generality, we assume that the prepared data w_i are converted into integer values and lie within the range of $[0, n-1]$ where n is a power of two. Note that we can easily extend the range so that the condition $n = 2^h$ holds. Continuing with our example, we can initialize $n = 2048 = 2^{11}$ to appropriately cover all possible return values w_i in the range $0 \leq w_i \leq 2048$.

5.2.2 Security Model

In our security model, all entities are assumed to be honest-but-curious participants, i.e., they faithfully follow the query processing protocols but might try and extract any additional query information during the data preparation and throughout the

query processing steps. For example, the fog node may be curious about each IoT device's data w_i and the user's query range $[L, U]$; each IoT device may be curious about other IoT devices' data and the query range $[L, U]$; and the query user may be curious about each IoT device data w_i, other than query response value Count($\mathbf{I}'$). We also assume there is no collusion between any two entities in our model.

Note that, an external adversary may launch active attacks on data integrity and source authentication, but as this work focuses on communication efficiency and privacy those attacks are beyond the scope of this study and left for future work.

5.2.3 Design Goals

Given the above system and security models, our design goals are to present a privacy-preserving and communication-efficient range query scheme for fog-based IoT. In particular:

• *The proposed scheme should be privacy-preserving.* Both the lower and upper bound values in the query $[L, U]$ should be private, i.e., no one, except the user, can retrieve $[L, U]$ to trace the query user's interests. In addition, the data collected by IoT devices are also sensitive and should be private, i.e., those data must only be accessible by their owners. Neither the fog node nor the query user is permitted to retrieve/recover the plaintext value w_i of each IoT device I_i.

• *The proposed scheme should be efficient.* Energy consumption is the critical bottleneck in IoT ecosystems and, at the same time, reaching the above privacy goal will introduce additional communication overhead. Therefore, in the proposed scheme, we aim to improve the communication efficiency by significantly reducing the number of encrypted blocks and achieving $O(\log^2 n)$ communication efficiency, much better than previously reported ones [85, 89].

5.3 Preliminaries - The XOR Gate

In order to accomplish the above communication-efficient privacy-preserving range query scheme in fog-based IoT, we need to adopt homomorphic encryption techniques, which support both homomorphic addition and multiplication. Since most of the existing homomorphic encryption schemes are in the public key settings and computationally inefficient, we leverage the SHE scheme [89], discussed in Chapter 4, as a fundamental building block for our privacy-preserving range query. In this section, we review the XOR operation and introduce our homomorphic XOR operator to check equality/inequality of encrypted bit values as it is the core operation of our proposed scheme. Note that the SHE and homomorphic schemes differ from the order revealing encryption (ORE) technique [72, 154, 139]. ORE schemes yield searchable encryption instead of homomorphic operations and aggregation. They allow range queries (e.g., MIN. MAX, and COUNT, but not AVG and SUM) on encrypted data, but would reveal the original data order and distribution.

One of the most useful digital logic gates to compare binary values is the XOR gate ($\oplus$), i.e., $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, and $1 \oplus 1 = 0$. In our proposed scheme, we need to homomorphically compare pairs of encrypted zeros and ones. Therefore, for the given SHE ciphertext values $E(X)$ and $E(Y)$, the following homomorphic expression must be evaluated to securely obtain $E(X \oplus Y)$.

$$E(X \oplus Y) = E(X) + E(Y) - 2E(X)E(Y) \tag{5.4}$$

5.4 Our Proposed PBTree-based Scheme

Our $O(\log^2 n)$ communication-efficient privacy-preserving range query scheme for fog-based IoT has roots in both a novel path reduction process and SHE homomorphic encryption. We first describe our path reduction process, which is executed

at the query user side to generate the encrypted reduced paths from a given range query $[L, U]$ over the range $[0, n-1]$. Then, we describe our privacy-preserving range query scheme in detail: 1) initializing the SHE cryptosystem (key generation phase); 2) generating encrypted range query at user side; 3) responding to the encrypted query at IoT device; 4) aggregating the encrypted responses in the fog node; and 5) decrypting the aggregated response to get the final result at user side.

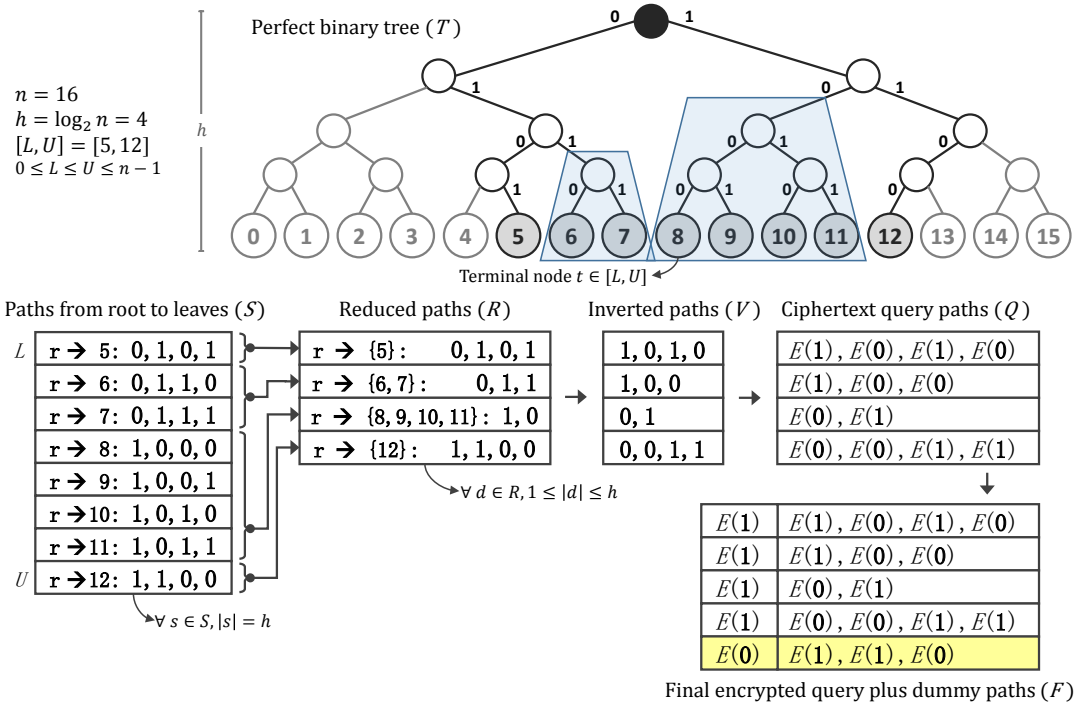


Figure 5.2: An example of converting a simple continuous range query into encrypted reduced paths

5.4.1 Range Queries as Reduced Paths

Our $O(\log^2 n)$ communication-efficient algorithm formulates the range query $[L, U]$, where $0 \le L \le U \le n-1$ and n is a power of two, as a reduced path structure in which the length of each entry lies between 1 and $h = \log_2 n$. This process consists of the following steps:

5.4.1.1 Construct a Perfect Binary Tree (PBTree)

Given a range query $[L, U]$ over $[0, n-1]$, the perfect binary tree T is constructed in which the leaf nodes are sequentially labeled from left to right with numbers from zero to $n-1$. Moreover, the left edges of T are labeled by zeros and the right edges by ones. For example, Fig. 5.2 depicts the sample perfect binary tree T for $n = 16$ and simple continuous range $[L, U] = [5, 12]$.

Notice that this scheme, unlike [85] and [89], supports not only simple continuous range queries $[L, U]$, but also non-continuous range queries which consist of two or more non-adjacent range queries, i.e., $[L, U] = \cup_{\iota=1}^{s}[L_\iota, U_\iota] = [L_1, U_1] \cup [L_2, U_2] \cup \cdots \cup [L_s, U_s]$. For example, a non-continuous range query and its corresponding reduced paths are depicted in Fig. 5.3.

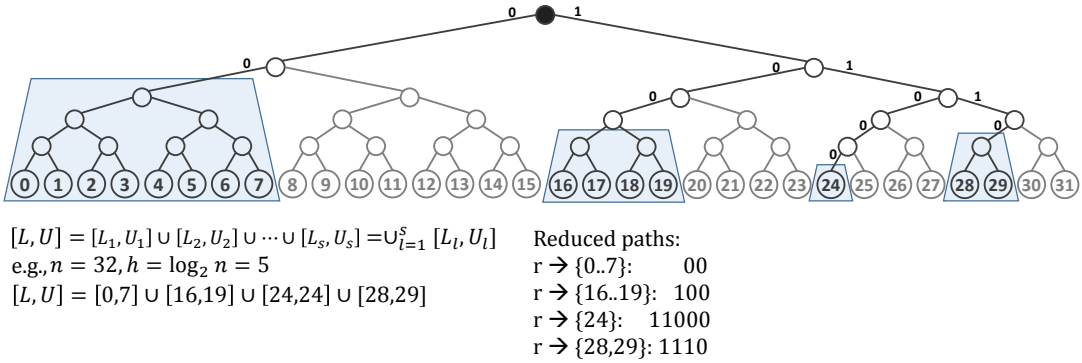


Figure 5.3: Non-continuous range query consists of two or more individual ranges and its corresponding reduced paths

Definition 5.4.1. *(PBTree): A perfect binary tree T with n leaf nodes has $n-1$ remaining non-terminal nodes, including a root node and $n-2$ internal nodes. It can be declared as below in which all non-terminal nodes have both left and right non-empty subtrees, and all leaves have the same height $h = \log_2 n$, where $n \geq 2$ is a power of two. In addition, edges leading to the left and right subtrees are, respectively, labeled with zeros and ones.*

```
public class PBTree {
```

```
    Node root;
    static class Node {
        int data;
        Node left, right;
        Node(int data) {
            this.data = data;
            this.left = null;
            this.right = null;
        }
}
```

5.4.1.2 Range Query Encoding

In this step, terminal node t in range query $[L, U]$ is encoded into binary-string s that represents the path from the root to the terminal node t. To this end, Algorithm 7 recursively traverses from root to all leaf node t in $[L, U] = \cup_{\iota=1}^{s}[L_\iota, U_\iota]$ and uses the visited array to keep track of the discovered vertices. Therefore, at the end of this traversal process, there exists the entry $s \in S$ that indicates the corresponding path from root to leaf nodes t in $[L, U]$. Moreover, the size of S is $U - L + 1$, i.e., $|S| = U - L + 1$. Also, the length of each entry in S equals the height of PBTree T, i.e., $|s \in S| = \log_2 n = h$. For example, in Fig. 5.2, the path from the root to terminal node $t = 11 \in [5, 12]$ can be expressed as "$\mathtt{r}, \mathtt{l}, \mathtt{r}, \mathtt{r}$", equivalent to $s =$ "1011" by simply replacing "$\mathtt{r}_{(right)}$"s with "1"s and "$\mathtt{l}_{(left)}$" with "0". Also, note that the length of each entry $s \in S$ is $|s| = h = \log_2 16 = 4$.

5.4.1.3 Reducing Paths

To achieve $O(\log^2 n)$ communication efficiency, our path reduction algorithm combines the listed paths in S and forms the reduced paths R. Fig. 5.2 illustrates an example of our path reduction technique output and Algorithm 8 describes in detail how to merge the distinct paths to form R from S. In each iteration of the loop in

Algorithm 7 Range Query to Traversal Path Strings

Input: PBTree T, $[L, U] = \cup_{\iota=1}^{s}[L_\iota, U_\iota]$; $0 \leq L \leq U \leq n-1$
Output: ArrayList S

```
                                  # Paths from root to leaf terminal nodes that belong to the range query
 1: PathToLeaf ← Ø                # ArrayList<Character> PathToLeaf
 2: S ← Ø                         # ArrayList<String> S
 3: for each t ∈ [L, U] do
 4:     T.GetPath(T.root, PathToLeaf, t, ' ')
 5:     S.Add(PathToLeaf)
 6:     PathToLeaf.Clear()
 7: return S

 8: procedure GetPath(RT, PTNode, DNode, VEdge)
                                  # RT=root, PTNode=PathToNode,
                                  # DNode=DestinationNode, VEdge=VisitedEdge
 9:     if RT = null then
10:         return false
11:     PTNode.Add(VEdge)        # Accumulate the visited edges
12:     if RT.data = DNode then
13:         return true
14:     if GetPath(RT.left, PTNode, DNode, '0') ∨ GetPath(RT.right, PTNode, DNode, '1') then
                                  # 0 for traversing left subtrees and 1 for right ones
15:         return true
16:     PTNode.Remove(PTNode.Size() − 1)
17:     return false
```

our reduction algorithm, two leaf nodes or sibling subtrees with corresponding path strings $\mathcal{P}$ and $\mathcal{P}'$ are reduced if (i) they are both of the same length (ℓ), (ii) their prefix substrings of length $\ell-1$ are equal and (iii) they differ only in their last entry. This process continues until reaching the maximum possible pruning and stops when there are no more adjacent path strings to be reduced. ($\mathcal{P}$ and $\mathcal{P}'$ are zero-based indices.)

$$(|\mathcal{P}| = |\mathcal{P}'| = \ell) \wedge (\mathcal{P}_{[0..\ell-2]} = \mathcal{P}'_{[0..\ell-2]}) \wedge (\mathcal{P}_{[\ell-1]} = \mathcal{P}'_{[\ell-1]}) \tag{5.5}$$

For example, the path from root to two different leaf nodes five ("0101") and six ("0110") are not reducible due to their different prefix substrings of length $\ell-1=3$. However, as shown in the figure, the two subtree sibling nodes six ("0110") and seven ("0111") are reducible and they will be merged into single path string "011" as they follow our reduction rule.

5.4.1.4 Inverting Reduced Paths

Since the XOR operation will eventually be applied to check whether an IoT device's prepared data is in the range query or not, the reduced paths R are flipped to get the inverted paths V. As shown in Fig. 5.2, inverted paths V are simply obtained by flipping zeros to ones and ones to zeros in each string entry of R.

5.4.1.5 Encrypting Inverted Paths

Inverted paths V, in turn, are encrypted to form the ciphertext query paths Q. To do so, we simply encrypt zeros and ones by using the SHE scheme.

5.4.1.6 Adding Dummy Paths

Let us consider what happens when the query user examines the range query: $[L, U] = [0, \frac{n}{2} - 1]$ (namely the entire left subtree). The reduced path, in this case, is "0" and consequently the encrypted inverted path contains just one entry including

Algorithm 8 Reducing Paths

Input: ArrayList S, $|s \in S| = h$
Paths from root to leaf nodes

Output: ArrayList R, $1 \leq |d \in R| \leq h$
Reduced paths

```
1: ArrayList RPath[2]                    # RPath[0] and RPath[1]
2: RPath[0] ← S
3: Src ← 0, Dst ← 1                      # Determine the source and destination RPath
4: flag ← false                          # Terminate the loop execution
5: repeat
6:     flag ← false
7:     for each d_i ∈ RPath[Src] do
                                         # 0 ≤ i < RPath[Src].Size()
8:         if i < RPath[Src].Size() − 1 then
9:             strRes ← Reduce(RPath[Src].Get(i), RPath[Src].Get(i + 1))
10:            if strRes = "" then
11:                RPath[Dst].Add(RPath[Src].Get(i))
12:            else
13:                RPath[Dst].Add(strRes)
14:                flag ← true
15:                i++
16:        else
17:            RPath[Dst].Add(RPath[Src].Get(RPath[Src].Size() − 1))
18:    if flag = true then               # Plan for the next iteration
19:        RPath[Src].Clear()
20:        Src = (Src +1) mod 2
21:        Dst = (Dst +1) mod 2
22: until flag = true
23: R ← RPath[Dst]
24: return R

25: procedure Reduce(x, y)
26:    ℓ_x ← x.Length()
27:    ℓ_y ← y.Length()
28:    if ((ℓ_x = ℓ_y) ∧ (x_[0..ℓ_x−2] = y_[0..ℓ_y−2]) ∧ (x_[ℓ_x−1] ≠ y_[ℓ_y−1])) then
29:        return x_[0..ℓ_x−2]
30:    return ""
```

one encrypted value, i.e., "$E(1)$". However, transmitting it to the fog node would lead to a high probability of inferring the query. For instance, in our above example, by simply submitting ciphertext "$E(1)$", the fog node can infer that the query user has examined one of the following queries: the left subtree ($[L, U] = [0, \frac{n}{2} - 1]$) or right subtree ($[L, U] = [\frac{n}{2}, n - 1]$). To overcome this problem, dummy paths (shaded yellow row in Fig. 5.2) are injected to prevent the curious fog node from inferring or guessing the exact range query. They are appended to encrypted query paths Q to output final encrypted query F. The theoretical analysis on dummy paths' length and guessing probability is detailed later in Section 5.5. Lastly, an additional encrypted column is added to each path entry to distinguish the dummy paths from the real ones. $E(1)$ will be assigned to valid paths and $E(0)$ to dummy paths. Therefore, since the dummy paths are multiplied by $E(0)$, they will be implicitly discarded during the aggregation phase in the fog node.

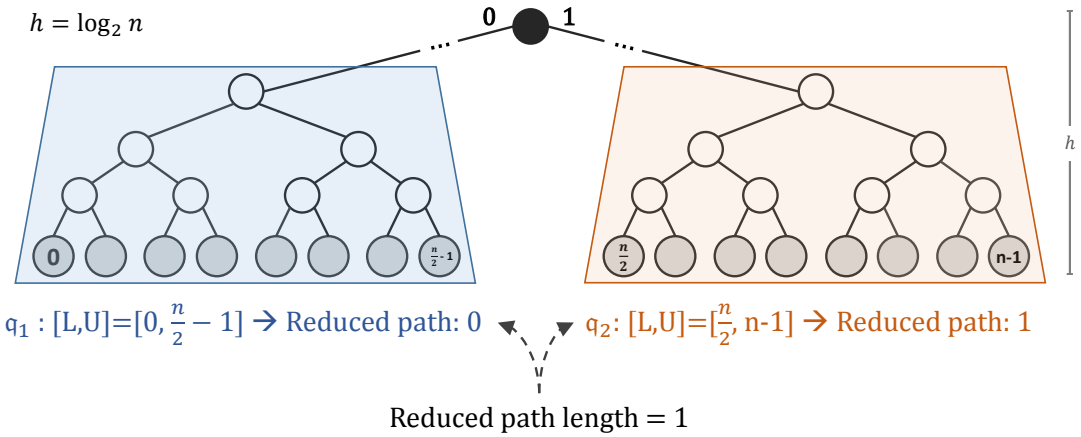


Figure 5.4: Lower bound communication complexity (minimum reduced path length) with $\mathfrak{q}_1$ and $\mathfrak{q}_2$ that have respectively queried left and right subtrees

5.4.2 Communication Complexity Analysis

The communication complexity of the proposed scheme is influenced by the total number of encrypted zeros and ones that are transmitted from the query user to the

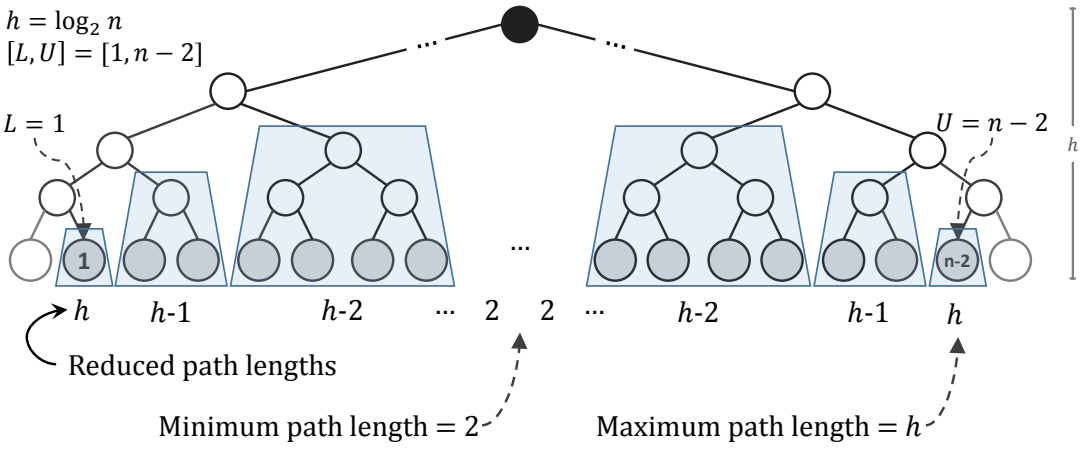


Figure 5.5: Upper bound communication complexity in $[L, U] = [1, n-2]$

fog node and IoT nodes. We now examine the minimum and maximum number of encrypted blocks to obtain the lower and upper bounds of the communication cost.

Lower bound analysis: Consider a user's range query when the query is covering exactly half of the whole domain (as depicted in Fig. 5.4), namely the left subtree of T ($[L, U] = [0, \frac{n}{2} - 1]$) or the right one ($[L, U] = [\frac{n}{2}, n-1]$). The length of the reduced path in both cases is one. Thus the lower bound is achieved and the communication complexity is $O(1)$.

Upper bound analysis: The upper-bound value can be obtained by issuing the range query: $[L, U] = [1, n-2]$ (as shown in Fig. 5.5) in which the maximum number of encrypted blocks equals $(h-1)(h+2)$ where $h = \log_2 n$. Hence, the upper bound communication complexity is $O(\log^2 n)$.

$$\begin{aligned}\text{Upper Bound} : & \; h + (h-1) + \cdots + 2 + 2 + \cdots + (h-1) + h \\ & = 2\Big(h + (h-1) + (h-2) + \cdots + 2\Big) \\ & = (h-1)(h+2) \in O(\log^2 n)\end{aligned}$$

5.4.3 Description of Our Proposed Scheme

In this subsection, we give a detailed description of the steps involved in our scheme.

5.4.3.1 Query User Key Generation

For simplicity, we consider $n = 2^h$ to represent n as an h-bit binary value. Given the input parameters k_0, k_1, k_2, the query user generates a secret key $SK = (p, q, \mathcal{L})$ and a public parameter $PP = (k_0, k_1, k_2, \mathcal{N})$ for the SHE scheme. Then, the query user keeps the secret key SK secretly and publishes the public parameter PP to the fog node and all IoT devices. It should be noted that achieving $O(\log^2 n)$ communication efficiency requires the following further assumptions:

- Message space depends on k_1 and it should be set to at least $\lceil \log_2 N \rceil$, where N is the number of IoT devices, as it needs to cover the maximum value of the count range query $\text{COUNT}(\mathbf{I}') = |\mathbf{I}'|$, i.e. $|\mathbf{I}| = N$.

- To achieve the intended communication efficiency, the SHE scheme should accept at least $2h$ homomorphic multiplication (h multiplication during XOR operation, $h - 1$ for multiplying the XORs' outputs, and one multiplication through applying a filter to clear out dummy paths), i.e., k_0 should be set equal to $2(2h+1)k_2$ in order to successfully evaluate the $2h$-depth multiplicative circuit.

5.4.3.2 Range Query Generation at Query User

The range count query $\text{COUNT}(\mathbf{I}') = |\mathbf{I}'|$, where $\mathbf{I}' = \{I_i | I_i \in \mathbf{I} \wedge w_i \in [L, U]\}$ allows the user to retrieve the number of IoT devices I_i, whose data w_i are in the range $[L, U]$ where $0 \leq L \leq U \leq n - 1$. In order to fulfil the privacy-preserving range query, the following steps should be taken by the query user:

Step 1: Construct the perfect binary tree T of height $h = \log_2 n$, with n leaf nodes

labeled, from left to right, with $0, 1, \cdots, n-1$ and the left and right edges are labeled with zero and one, respectively.

Step 2: Generate the path from the root to leaves whose corresponding labels are in the range query, i.e., $t \in [L, U]$.

Step 3: Combine the path strings to obtain the reduced paths R. The length of each entry in R is between one and h, i.e.,$\forall\ d \in R, 1 \leq |d| \leq h$.

Step 4: Invert each reduced path string by replacing zeros by ones and vice versa.

Step 5: Encrypt zeros and ones (separately) in each inverted path string by using SHE.

Step 6: Append dummy encrypted paths (encrypted zeros and ones) to prevent the fog node from inferring the actual query, more likely when it is a special-case query.

Step 7: Add the encrypted filter as a new column to wipe out the dummy paths (additional column in the final encrypted query in Fig. 5.2). More specifically, $E(1)$s are assigned to real reduced paths, whereas $E(0)$s are assigned to dummy paths, i.e., $E(0)$s behave as a filter to eliminate the dummy paths during the homomorphic aggregation. After that, the encrypted range query F will be sent to all IoT devices via the fog node. To be precise, the encrypted range query F includes the real paths, dummy paths, and the filtering column, together with an additional pair of encrypted zero and one, i.e., $(E(0), E(1))$. Note that each IoT device I_i can formulate its encrypted response without knowing the secret key SK during the response generation phase based on the additional pair of $(E(0), E(1))$.

5.4.3.3 Query Response at IoT Devices

Upon receiving the range query F, including real and additional dummy path strings, each IoT device $I_i \in \mathbf{I}$ needs to generate the encrypted response as follows. Since SHE is a symmetric encryption scheme, and no one other than the query user knows the secret key SK, each IoT device I_i cannot directly encrypt its data w_i. Therefore,

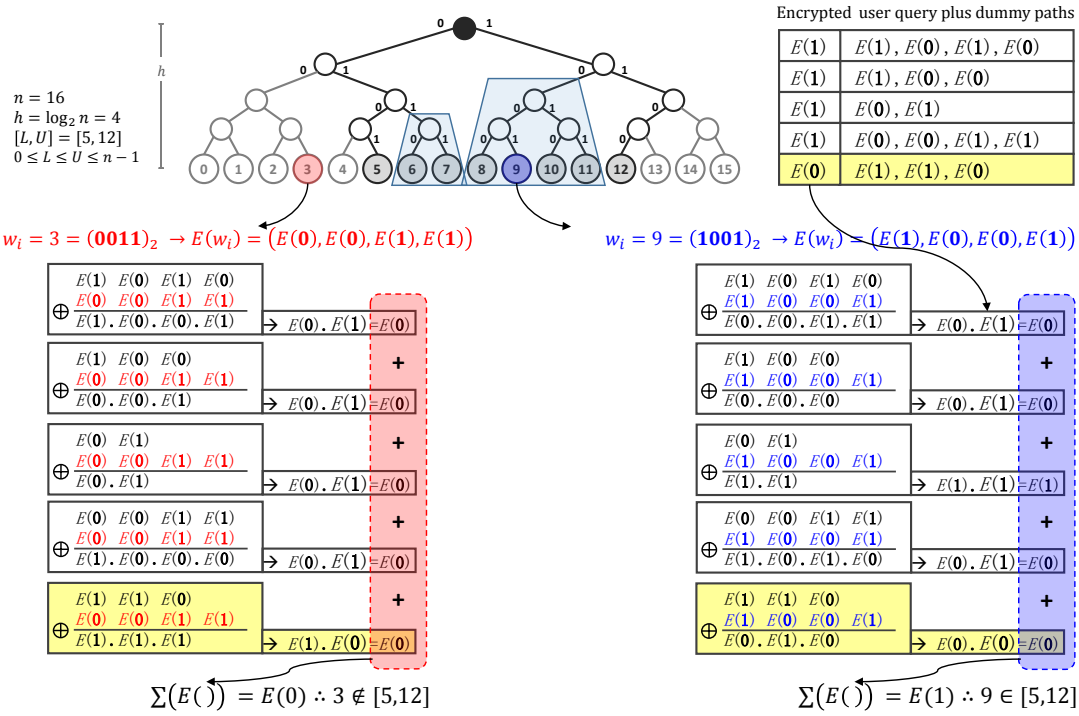


Figure 5.6: IoT device I_i's response for two different sensed data $w_i = 3$ and 9, where $3 \notin [5, 12]$ and $9 \in [5, 12]$

instead of encrypting w_i, each IoT device can form the encrypted equivalent data $E(w_i)$ by combining $E(1)$'s and $E(0)$'s from the additional pair $(E(0), E(1))$ in the range query F.

Step 1: I_i converts the sensed and prepared data w_i into binary form, where $0 \leq w_i \leq n-1$. This is similar to traversing the perfect binary tree with n leaves, from root to leaf node whose label is w_i. For example, as it is shown in Fig. 5.6, the path strings from root to leaves for two different sensed data 3 and 9 are "0011" and "1001", respectively.

Step 2: I_i generates the corresponding encrypted value for the binary representation of path string. I_i can generate the encrypted equivalent value for each sensed data w_i by combining $E(1)$'s and $E(0)$'s. For example, the encrypted equivalent value of "0011" and "1001" are $E(0011) = \Big(E(0), E(0), E(1), E(1)\Big)$ and $E(1001) = \Big(E(1), E(0), E(0), E(1)\Big)$.

Step 3: I_i performs the homomorphic XOR operations over all entries in the encrypted query F and previously computed $E(w_i)$ as listed in the large rectangles in Fig. 5.6.

Step 4: The output of the XORs are homomorphically multiplied to get the encrypted partial results. Then, these are multiplied by encrypted values in the supplementary filter column of F to clear out the dummy path strings from the response. For example, in the case of $w_i = 3$, our carefully devised filter column has nullified the effect of possible $E(1)$s from the response by setting the filter column to $E(0)$ for dummy path entries (yellow shaded rectangle while calculating the response for $w_i = 3$).

Step 5: The output of the previous step, in small rectangles, will homomorphically be added to each other to get the final encrypted response $Res[I_i]$.

$$\begin{aligned} & Res[I_i] \\ &= \sum_{\forall f \in F^{\text{real}}} E(1)\Big(\prod f \oplus w_i\Big) + \sum_{\forall f \in F^{\text{dummy}}} E(0)\Big(\prod f \oplus w_i\Big) \end{aligned}$$

Step 6: Finally, to protect $Res[I_i]$ from the semi-honest fog node, the IoT device I_i should self-blind the encrypted response $Res[I_i]$ by converting it into another valid ciphertext as follows:

$$Res[I_i] = Res[I_i] + \Big(E(0) \times \mathsf{r}\Big)$$

where $\mathsf{r} \in \mathcal{M}$. This will prevent the fog node from generating different combinations of $E(0)$ and $E(1)$ of IoT device I_i to guess the corresponding sensed data w_i. Eventually, I_i can send back the self-blinded ciphertext result $Res[I_i]$ to the fog node.

5.4.3.4 Response Aggregation at Fog Node

After receiving all response values $Res[I_i]$ from IoT devices, the fog node aggregates the encrypted responses to generate the final encrypted result $Count$ and forward it back to the user.

$$Count = \sum_{I_i \in \mathbf{I}} Res[I_i]$$

5.4.3.5 Response Recovery at Query User

Upon receiving the encrypted query result $Count$, the query user uses secret key SK to recover the range query response as:

$$Count \xrightarrow{D()} \text{Count}(\mathbf{I}') = |\mathbf{I}'| = \sum_{I_i \in \mathbf{I}} Res[I_i]$$

The correctness of the result is verified as below:

$$Count = \sum_{I_i \in \mathbf{I}} Res[I_i] = \sum_{I_i \in \mathbf{I}'} E(1) + \sum_{I_i \notin \mathbf{I}'} E(0) = E(|\mathbf{I}'|)$$

5.5 Security Analysis

In this section, we analyze the security of our proposed scheme. We start with preserving the privacy of range query $[L, U]$ and then continue with protecting the privacy of subset $\mathbf{I}'$.

• *The query range* $[L, U]$ *is privacy-preserving in our proposed scheme.* As discussed in Subsection 5.4.1, in order to achieve communication efficiency, the query range $[L, U]$ $(0 \leq L \leq U \leq n-1)$ is turned into encrypted reduced paths F with minimum 2 and maximum $(\log n - 1)(\log n + 2)$ encrypted blocks. Each encrypted entry in F has been encrypted by SHE, which has been proved to be secure under the known-plaintext attack [89]. SHE's security is based on the large integer factoriza-

tion problem (LIFP). Therefore, by properly setting the parameters, SHE is secure while LIFP is hard which means SHE guarantees that without knowing the secret key $SK = (p, q, \mathcal{L})$, the adversary has no idea of the entries inside F. Since neither the fog device nor IoT devices can access the secret key SK, they have no idea of the encrypted reduced path strings inside F, i.e., they cannot distinguish whether a ciphertext in F is encrypted from zero or one. Precisely, without plaintext information about entries in F, the fog node and IoT devices have no idea of the query range $[L, U]$ or even part of it.

Moreover, in the case of executing range queries with special values, e.g., $[L, U] = [0, n-1]$, as F has just two single encrypted entries, i.e., $E(0)$ and $E(1)$, there is a possibility of inferring the range query. Hence, to prevent the fog node and IoT devices from guessing the range query, dummy paths are appended to the real entries in F as well as additional filter column to distinguish the real entries from dummy ones. Adding dummy path(s) provides an effective approach to reducing the probability of guessing the encrypted queries, especially when the query path length is short. For example, in Fig. 5.7, the length of the ciphertext query path (Q) is $\mathfrak{L} = 2$. That would mean that the curious entity can guess the original query by selecting one of the four possible cases. i.e., "00", "01", "10", or "11". Extending to the more general form, consider the original range query with one entry of length $\mathfrak{L}$. In this case, without injecting dummy path(s) there are $2^{\mathfrak{L}}$ different possible queries. Therefore a curious entity can correctly guess the exact range query with probability $\frac{1}{2^{\mathfrak{L}}}$. Injecting one dummy path instance of length $\mathfrak{L}'$ limits the probability of a correct guess by the adversary in two ways. First, since a dummy path instance of length $\mathfrak{L}'$ is injected, it generates $2^{\mathfrak{L}'}$ additional cases. Second, for the adversary, both a dummy path instance and a real range query path are indistinguishable; therefore, the adversary needs to search the whole string path of length $\mathfrak{L} + \mathfrak{L}'$ with $2^{\mathfrak{L}+\mathfrak{L}'}$ new cases. Consequently, a curious entity needs to guess the exact query from

either dummy path instances ($2^{\mathfrak{L}'}$), real range query paths ($2^{\mathfrak{L}}$) or both ($2^{\mathfrak{L}+\mathfrak{L}'}$). This means that the probability of guessing the user's real range query will be bounded from $\frac{1}{2^{\mathfrak{L}}}$ to $\frac{1}{2^{\mathfrak{L}}+2^{\mathfrak{L}'}+2^{\mathfrak{L}+\mathfrak{L}'}}$. Besides, to interdict the curious entity from successfully guessing the real range query and to bind the probability less than or equal to ε, we need to set $\mathfrak{L}' \geq \lceil \log_2 \frac{\frac{1}{\varepsilon}-2^{\mathfrak{L}}}{1+2^{\mathfrak{L}}} \rceil$, which is given as follow:

$$\frac{1}{2^{\mathfrak{L}} + 2^{\mathfrak{L}'} + 2^{\mathfrak{L}+\mathfrak{L}'}} \leq \varepsilon$$
$$2^{\mathfrak{L}} + 2^{\mathfrak{L}'} + 2^{\mathfrak{L}+\mathfrak{L}'} \geq \frac{1}{\varepsilon}$$
$$2^{\mathfrak{L}'}(1 + 2^{\mathfrak{L}}) \geq \frac{1}{\varepsilon} - 2^{\mathfrak{L}}$$
$$\mathfrak{L}' \geq \lceil \log_2 \frac{\frac{1}{\varepsilon} - 2^{\mathfrak{L}}}{1 + 2^{\mathfrak{L}}} \rceil$$

For example, consider an original range query with an entry of length $\mathfrak{L} = 2$ in Fig. 5.7. In order to bind the probability of correctly guessing the range query from $\varepsilon = \frac{1}{2^{\mathfrak{L}}} = \frac{1}{4}$ to $\varepsilon = \frac{1}{50}$, we need to add dummy path(s) of length $\mathfrak{L}' \geq \lceil \log_2 \frac{\frac{1}{\varepsilon}-2^{\mathfrak{L}}}{1+2^{\mathfrak{L}}} \rceil = \lceil \log_2 \frac{50-2^2}{1+2^2} \rceil = 4$. Note that the dummy paths of length $\mathfrak{L}' > h$, where $h = \log_2 n$, need to be broken down into separate dummy path instances of maximum length h for each instance.

• *The subset $\mathbf{I}'$ is also privacy-preserving in the proposed scheme.* The subset $\mathbf{I}'$ denotes a set of IoT devices whose data are within the query range $[L, U]$, i.e., $\mathbf{I}' = \{I_i | I_i \in \mathbf{I} \wedge w_i \in [L, U]\}$. As described in the security model, $\mathbf{I}'$ should be kept secret from the query user, fog device, and each IoT device.

First, when the fog node's encrypted aggregated query response is received by the query user, he/she will decrypt using the secret key SK to obtain the final result. This means that the query user only knows the number of IoT devices whose data are within the requested query range $[L, U]$, but has no idea which specific IoT device has a data within the query range. Thus, the subset $\mathbf{I}'$ can be kept secret from the query user.

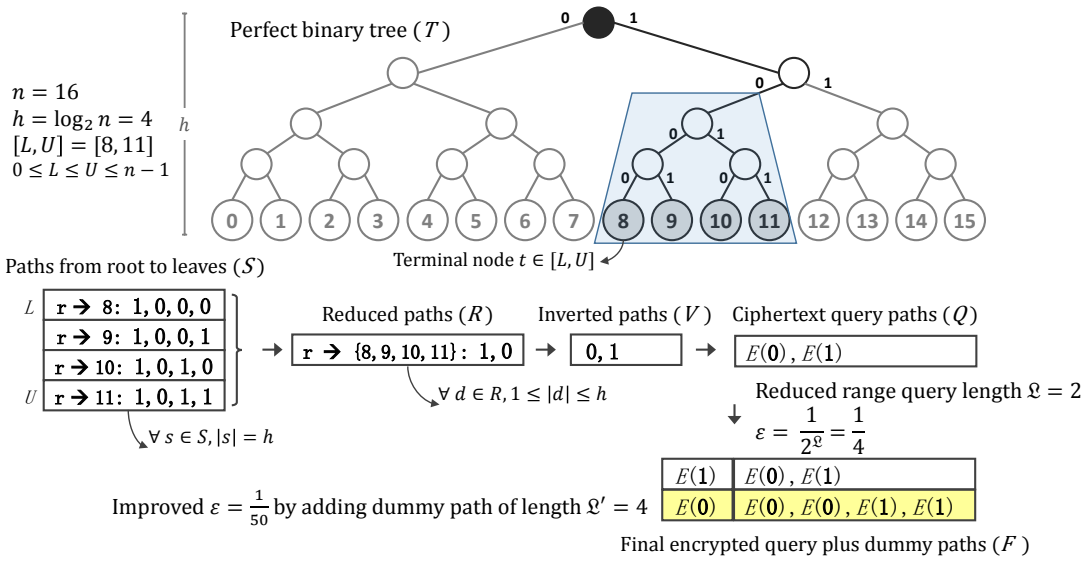


Figure 5.7: Adding dummy path instance of length $\mathfrak{L}' = 4$ to original range query of length $\mathfrak{L} = 2$ to bind the probability (ε) of correctly guessing the user's range query; $\varepsilon = \frac{1}{2^{\mathfrak{L}}+2^{\mathfrak{L}'}+2^{\mathfrak{L}+\mathfrak{L}'}}$

Second, on one hand, the fog node receives the encrypted query F from the query user and forwards it to the IoT devices; on the the other hand, the IoT devices' encrypted responses, i.e., $Res_{(I_i)}$s, will be aggregated at the fog node and the final ciphertext response $\sum_{I_i \in \mathbf{I}} Res[I_i]$ is forwarded back to the query user. Since the query range $[L, U]$ is transformed into encrypted query F by applying SHE, the security of SHE guarantees that the fog node has no idea on the plaintext of the query range. Then, the fog node has no way to determine the subset $\mathbf{I}'$ by observing encrypted path strings F. At the same time, each encrypted response $Res_{(I_i)}$ is self-blinded with random factor $\Big(E(0) \times \mathsf{r}\Big)$ to make the brute-force attack impossible. This would mean that if there is no self-blinding factor $\Big(E(0) \times \mathsf{r}\Big)$, it would be possible for the fog node to identify the specific IoT device's response, i.e., it can iterate over all possible prepared values w_i $(0 \le w_i \le n-1)$ and compute the corresponding response for each entry in F to check whether $Res_{(I_i)} \stackrel{?}{=} \sum_{\forall f \in F^{\text{real}}} E(1)\Big(\prod f \oplus w_i\Big) + \sum_{\forall f \in F^{\text{dummy}}} E(0)\Big(\prod f \oplus w_i\Big)$. Although this attack requires a lot of computations, it is feasible, especially when n is small value. Luckily, protecting $Res[I_i]$ with

$\Big(E(0) \times \mathsf{r}\Big)$ will make it impossible for the fog node to launch a brute force attack on an individual IoT device's response. Therefore, it is impossible for the fog node to obtain the information about the subset $\mathbf{I}'$ and the subset $\mathbf{I}'$ is kept secret from the fog node.

Third, for each IoT device I_i, if it attempts to obtain the information about $\mathbf{I}'$, it needs to determine whether itself is in $\mathbf{I}'$ or not and whether other IoT devices are in $\mathbf{I}'$ or not. As discussed before, I_i uses its prepared data w_i to compute the encrypted $Res[I_i]$, but has no idea on $Res[I_i]$ is $E(0)$ or $E(1)$. In other words, I_i does not know whether w_i is in the query range $[L, U]$ and I_i is in $\mathbf{I}'$. Besides, since I_i cannot access the plaintext data prepared by the other IoT devices and the plaintext of query range, it has no idea whether the other IoT devices are in $\mathbf{I}'$ or not. Thus, the subset $\mathbf{I}'$ is kept secret from other IoT devices.

5.6 Performance Evaluation

In this section, we give the result of an experimental assessment to analyze the effectiveness of the proposed scheme with respect to the computational costs and communication overheads. The detailed results confirm the practical performance of the scheme. Each step of the privacy-preserving range query in this scheme is compared with the corresponding steps in both previously proposed schemes, i.e., Lu's scheme [85] and the STSM scheme [89]. Lu's privacy-preserving range query, with $O(\sqrt{n})$ communication overhead, takes advantage of $\sqrt{n} \times \sqrt{n}$ matrix decomposition technique and is performed based on the BGN homomorphic cryptosystem [21]. The STSM scheme has a $O(\log^3 n)$ communication cost as it benefits from SHE as well as an encoding method to convert a given range query into $\log n + 1$ semi-triangular sparse matrices. The platform used to compare the schemes was an Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz, with 8 GB main memory running

Linux (Ubuntu 16.04). The detailed parameter settings for both schemes and all experiments are listed in Table 5.2.

Table 5.2: The parameter settings

Shared param.	Value
N	The number of IoT devices, $N = 1000$
n	The maximum sensed value, $n = \{2^6, 2^8, 2^{10}, 2^{12}, 2^{14}, 2^{16}, 2^{18}, 2^{20}, 2^{22}, 2^{24}, 2^{26}, 2^{28}, 2^{30}\}$
$[L, U]$	The range query lower and upper bound, $0 \leq L \leq U \leq n-1$
w_i	IoT device I_i's prepared data, randomly picked from $[0, n-1]$. i.e., $0 \leq w_i \leq n-1$

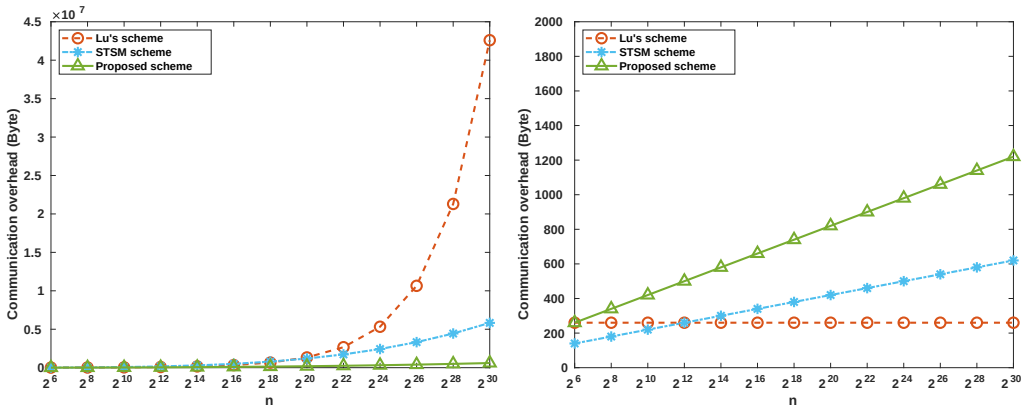
STSM and Our proposed scheme - SHE	
Parameter	Value
h	$h = \log_2 n$, i.e., $b = \{6, 8, 10, \cdots, 30\}$
k_0, k_1, k_2	$k_1 = \log_2 N$, $k_2 = 40$
k_0	$2(h+1)k_2$ in STSM, $2(2h+1)k_2$ in ours
p, q ,L, $\mathcal{N}$	$\lvert p\rvert = \lvert q\rvert = k_0$, $\lvert L\rvert = k_2$, $\mathcal{N} = pq$
$\mathcal{M}$	The query result $\mathcal{M} = N \in \{0,1\}^{k_1}$

Lu's scheme based - BGN	
Parameter	Value
h	$h = \sqrt{n}$, i.e., $h = \{2^3, 2^4, 2^5, \cdots, 2^{15}\}$
κ	$\kappa = 512$
p, q, $\mathcal{N}$	$\lvert p\rvert = \lvert q\rvert = \kappa = 512$, $\mathcal{N} = pq$
Δ	The query result's upper bound, $\Delta = N$

5.6.1 Communication Overhead

For the communication overhead analysis, we report the required overhead in both directions between the user and the fog node, i.e., the volume of encrypted data that is uploaded from the query user to the fog node and the single ciphertext response length from the fog node to the query user. First, Fig. 5.8(a) compares the communication overhead of transferring an encrypted query from the query user to the fog node, with n varying from 2^{10} to 2^{30}. It is clear that the overhead of the Lu's scheme grows tremendously as n increases. This is mainly caused by the simultaneous increase in the number of encrypted elements. Our proposed scheme is also more efficient than the STSM scheme. Our $O(\log^2 n)$ communication pattern will make more efficient use of bandwidth as compared to Lu's scheme and the STSM scheme with $O(\sqrt{n})$ and $O(\log^3 n)$, respectively. For instance, in case of $n = 2^{30}$, the communication cost in our proposed scheme is less than one MB comparing to almost 40.6 and 5.5 MB in Lu's scheme and the STSM scheme, respectively. Therefore, the communication cost from the query user to the fog node in our proposed scheme is remarkably low.

Secondly, Fig. 5.8(b) depicts the encrypted response length from a fog node to the query user. All these three schemes are efficient and with responses being less than one KB. However, unlike the STSM scheme and our proposed scheme, Lu's scheme is irrelevant to n. This is due to the different values for k_0 that should be set to $2(2h+1)k_2$ and $2(h+1)k_2$ in our proposed scheme and the STSM scheme to support $2h$ and h-depth homomorphic multiplications, respectively. In Lu's scheme, it is based on the BGN, ciphertext length is constant and depends on the security parameter κ. For example, in the case of $\kappa = 512$, the BGN ciphertext size is $4\kappa = 256$ Bytes or 0.25 KB.



(a) Requests from user to fog node (b) Responses from fog node to user

Figure 5.8: Communication overhead comparison (linear scale) between the proposed scheme and Lu's scheme with respect to n

5.6.2 Computational Cost

As well, in all three schemes, the computational costs are influenced by the number of encrypted elements that are generated by the query user during the range query encoding phase. Fig. 5.9(a) compares the number of encrypted in all three schemes. The computational costs of Lu's scheme and the STSM scheme increase rapidly with n. The difference is rooted in the decomposition method. Lu's scheme and the STSM scheme use $O(\sqrt{n})$ and $O(\log^3 n)$ decomposition approaches, respectively, whereas

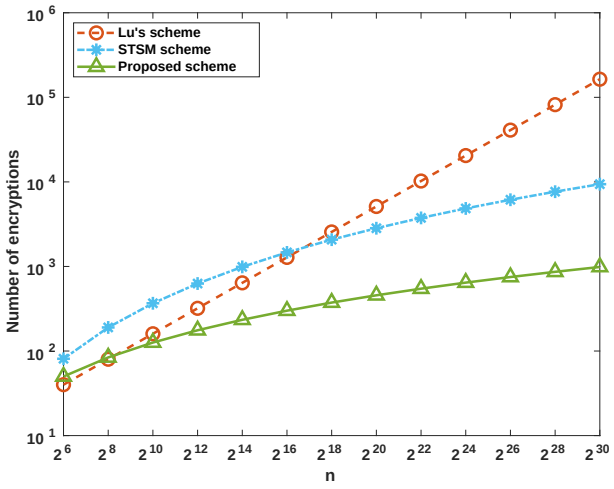
ours takes advantage of a faster and more efficient decomposition technique with $O(\log^2 n)$ communication efficiency.

Note: *Execution times for Lu's scheme are very high compared with both the STSM scheme and our proposed scheme (almost one millisecond in three out of the four sub-figures). We have therefore used a logarithmic scale to make interpretation easier.*

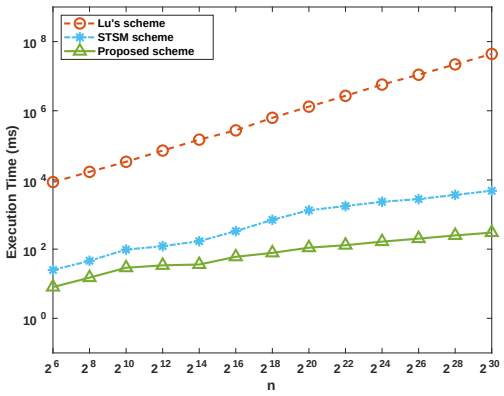
Fig. 5.9(b) shows the consuming time for query generation by the user. From the figure, we see that the computational cost of Lu's scheme dramatically grows with n, while the STSM and our proposed schemes have a significantly lower growth rate. We also observe that our scheme has a better execution time than the STSM scheme. The reason behind this is that the schemes are producing a different number of encrypted elements as discussed before. Therefore our proposed scheme achieves better results than the others in terms of query generation time.

Fig. 5.9(c) illustrates the response time for an IoT device. Although the IoT device response time in Lu's scheme is unaffected by problem size, it is still considerably higher than the other two. Additionally, we observe that our proposed scheme requires more computational time than the STSM scheme, as the IoT device's response generation needs more computation, i.e., a combination of XOR operations as well as more homomorphic additions and multiplications than the STSM scheme. However, the response time in both schemes is much less than one second. Putting these two results together, our proposed scheme's query generation time is 16 times faster than that of the STSM scheme, though the response time for each IoT device in our scheme is slightly longer than that of the STSM scheme. As a result, the overall execution time in our proposed scheme is better than the STSM scheme.

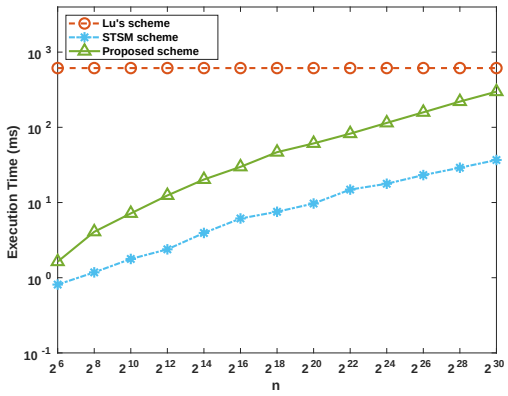
Fig. 5.9(d) depicts the response aggregation time at the fog node. It is mainly dependent on the number of IoT devices, i.e. N. Therefore, as it is seen from the figure, the aggregation time in all three schemes is independent of the problem size, i.e., n, even if there is a slight increase in both our proposed and the STSM



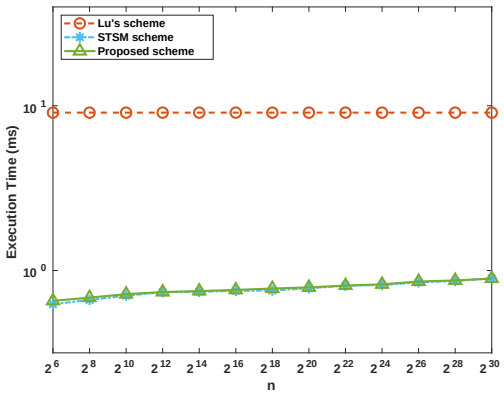
(a) The number of encryptions in the decomposition phase



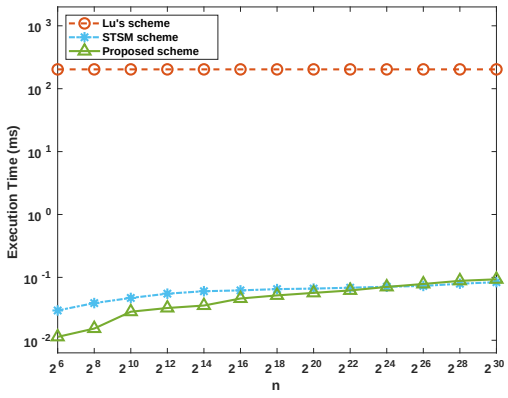
(b) Query generation at the user side



(c) Query response at the IoT device



(d) Response aggregation at the fog node



(e) Response recovery at the user side

Figure 5.9: The computational cost comparison between our scheme and Lu's scheme for n varying from 2^{10} to 2^{30}

schemes, varying with n. This is because of the gradually increasing ciphertext sizes with respect to problem size. In Lu's scheme the cipthertext length depends only on the security parameter κ and is consequently independent of n. Moreover, the aggregation times in both our proposal and the STSM scheme are almost the same as both schemes employ the SHE cryptosystem.

Finally, Fig. 5.9(e) presents the response recovery time which measures how much time the user spends decrypting the ciphertext upon receiving the encrypted response from the fog node. Obviously, they are still efficient and independent of problem size n. However, decryption in both the proposed and the STSM schemes (with a time less than one millisecond even for $n = 2^{30}$) is faster than Lu's scheme. The reason behind this is that the latter performs in the BGN scheme and over $\mathbb{G}_T$ as well as requiring relatively smaller message space than $\Delta = 1000$ to operate more effectively. Message decryption in the BGN scheme involves solving the discrete logarithm using Pollard's lambda method, i.e., the BGN scheme is time-consuming and less practical for larger message spaces.

It should also be noted that, in addition to improving significant portions of the query processing systems, our proposed scheme has addressed two limitations of the other two schemes, namely 1) ours does not require the lower and upper bound values be a power of two whereas the STSM scheme does, and 2) ours supports non-continuous range queries while theirs do not.

5.7 Related Work

In this section, we will take a brief look at some previously reported studies in privacy-preserving schemes for fog-based IoT applications. Most recently, authors in [89] have devised the SHE scheme along with decomposition technique to transform the range queries into semi-triangular sparse matrices to achieve $O(\log^3 n)$ com-

munication efficiency. Although the STSM scheme has acceptable communication and computation efficiency, it only accepts continuous range queries in the form of $[L = 2^a, U = 2^{a'}]$, i.e., the lower and upper bound values should be chosen as a power of two. Prior to this study, a privacy-preserving range query in fog-based IoT was studied in [85], which is another related study to our proposed scheme. The scheme proposed in [85] is built upon the BGN homomorphic encryption together with a range query expression, decomposition, and composition technique, that can achieve $O(\sqrt{n})$ communication efficiency. It is less efficient than the STSM scheme, but it accepts any continuous form of range queries. In addition, because the scheme is based on the BGN scheme with time-consuming bilinear pairing operations in public key settings, the computational cost is large. Aiming at improving the communication efficiency as well as resolving the deficiencies in the queries' form, our proposed scheme comes with a novel range query encoding technique to achieve $O(\log^2 n)$ communication-efficiency and overcome the weakness of the query format.

When the privacy-preserving range query on "*the number of IoT devices whose data x_i is within the range [L, U]*" is considered as a special privacy-preserving data aggregation scheme in fog-based IoT, there are other studies close to our proposed scheme. In [86], Lu et al. addressed heterogeneous data aggregation in real IoT applications by proposing a light-weight privacy-preserving data aggregation (LPDA) for a fog-enabled setting. The proposed LPDA is characterized by applying Paillier cryptosystem, the Chinese Remainder Theorem, and one-way hash chain function to aggregate hybrid IoT devices' data and to early filter the injected false data at the network edge. In [60], Huang et al. studied the fog-assisted selective aggregation operation. Specifically, they constructed a new threat model to formalize the non-collusive and collusive attacks of compromised fog nodes. Their proposed privacy-preserving and reliable selective multi-source aggregation scheme is comprised of the BCP cryptosystem, randomized message-lock encryption, homomorphic proxy-authenticators, and

multi-dimensional aggregation and can well tackle the data privacy and reliability challenges. Recently, Mahdikhani et al. [90] presented a privacy-preserving subset aggregation scheme in fog-enhanced IoT scenarios, which enables a query user to gain the sum of the prepared data from a subset of IoT devices. To identify the subset, inner product similarity of the normalized vectors in the query user side and each IoT device is securely computed. Only when the inner product is greater than the user's specified threshold, an IoT device's data will be privately aggregated to form the final response.

5.8 Concluding Remarks

In this chapter, we have proposed a novel privacy-preserving range query encoding technique based on SHE homomorphic encryption [89] and reduced paths concept to privately and efficiently execute range queries with $O(\log^2 n)$ communication efficiency. Compared with previously discussed studies [85, 89], our proposed scheme can also support non-continuous range queries for any given value of L and U. Also, extensive experimental results demonstrate that our proposed scheme shows significant improvements in both communication overhead and computational cost. Consequently, our proposed scheme is more efficient privacy-preserving range query in fog-based IoT environments. In the next chapter, we conclude this thesis and discuss possible directions for the future work.

Chapter 6

Conclusion and Future Work

In this chapter, we will first summarize our contributions in this dissertation towards privacy-preserving range query scheme in fog-based IoT. Then, we discuss possible extensions and directions for future research work.

6.1 Contributions

The major contributions of this dissertation can be listed as follows:

- First, we proposed our communication-efficient privacy-preserving range query in the fog-enhanced IoT. Our novel scheme was characterized by employing the Paillier homomorphic cryptosystem and ingenious Bloom filter data structure. It simultaneously satisfies both privacy and efficiency in achieving the count aggregate range query. More precisely, $(n + |E|) \log n$-bit communication efficiency could be achieved by our proposed scheme. Detailed security analysis showed that the scheme has achieved privacy preservation in the range query. Moreover, experiments were conducted and the results demonstrated the efficiency of the proposed solution compared with the existing BGN-based range query [85]. Unlike Lu's methods in which IoT end-devices need to respond to fog node by generating encrypted BGN's ciphertexts, our scheme calculating

the hash value of the prepared sensed data and Diffie-Hellman shared authentication key.

- Second, we design a new privacy-preserving range query scheme based on semi-triangular sparse matrices (STSM). Specifically, we first devise an efficient symmetric homomorphic encryption (SHE) scheme for maintaining data privacy and security in a range query. Then, we present a novel range decomposition technique to transform the range query, which can transform a given range query $[L, U]$, where $0 \leq L \leq U \leq n-1$, into a semi-triangular structure, and enable our proposed scheme to achieve $O(\log^3 n)$ communication efficiency. Although shorter response latency and lower bandwidth usage are made possible thanks to the lightweight SHE cryptosystem, our proposed scheme suffers from handling the flexible form of the range queries. More precisely, STSM accepts only continuous range queries where both lower and upper bound range values are in the form of two's power, i.e., $0 \leq L = 2^a \leq U = 2^{a'} \leq n-1$.

- Third, although communication overhead and computational overhead has been widely investigated in our previous solutions, a new efficient and privacy-preserving range query scheme in fog-enabled IoT is proposed to achieve more efficient and flexible scheme. The introduced scheme is based on reduced path decomposition/composition method. The decomposition technique interprets a given range query $[L, U]$, where $0 \leq L \leq U \leq n-1$, in a form of inverted reduced path strings. Then, the symmetric homomorphic encryption (SHE) scheme is employed to encrypt the reduced paths and hand them over securely through a fog node to the IoT devices. This technique enables a query user to launch a privacy-preserving continuous or non-continuous range query and receive a homomorphically aggregated encrypted response with an improved $O(\log^2 n)$ communication efficiency. Detailed security analysis showed that our proposed scheme is privacy-preserving. Besides, extensive performance

evaluations were also conducted, and the results demonstrated that our proposed scheme is by far more efficient than those previously reported schemes in terms of computational overhead and communication complexity. The proposed scheme supports both continuous and non-continuous range queries. Additionally, it accepts a flexible form of range queries than the STSM in which the lower and upper bound values should be in a limited format.

Table 6.1 is briefly summarized the property of the discussed schemes in terms of communication and computation complexities, types and kinds of supporting range queries, and underlying cryptographic algorithms.

.

6.2 Future work

Since key characteristics of fog have made it a suitable computing paradigm for new proliferating IoT devices, there still exist several open research challenges that are worth exploring in the future. We highlight only the most important ones, from our point of view.

- Proposing more computationally efficient schemes with reduced communication cost and computational overhead than the currently proposed schemes. These improvements can be made when ingenious decomposition/composition techniques are developed and applied to encrypted queries and responses.

- Supporting a variety of aggregate queries by suggesting new schemes to accept other types of aggregate functions than we have ever studied, e.g., min and max. Or proposing schemes with the ability to accept more aggregate functions within a single round-trip query instead of multi round-trip range queries.

- Launching more complex configurations to improve the proposed query pro-

Table 6.1: Comparison of the privacy-preserving aggregate range query schemes in fog-based IoT environment

Scheme	Accepting types of queries	Supporting range queries	Cryptographic alg.
Lu's scheme - $O(\sqrt{n})$	Only continuous	Count and sum Aggr.	BGN
Bloom filter-based scheme - $(n + \|E_{Paillier}\| + \|C\|) \log n$	Continuous and non-continuous	Only count Aggr.	Paillier, DiffieHellman, and SHA3-256
STSM scheme - $O(\log^3 n)$	Only continuous and queries in the form of $[L = 2^a, U = 2^{a'}]$	Count and sum Aggr.	Our proposed SHE
PBTree-based scheme -$O(\log^2 n)$	Continuous and non-continuous	Count and sum Aggr.	Our proposed SHE

continue	Preparation (btw. User → IoT)	User → Fog	Fog → IoT	IoT → Fog	Fog → User
Lu's scheme	-	$5\sqrt{n}(\|E_{BGN}\|)$	Same as User → Fog	$\|E_{BGN}\|$	$\|E_{BGN}\|$
Bloom filter-based scheme	- Cryptographic hash function (H) - Large prime number $p(\|p\|$=2048) - Primitive root g	$(n + \|E_{Paillier}\| + \|C\|) \log n$	$g^a \bmod p$, $\|p\|$=2048	$\|SHA3-256\|$	$\|E_{Paillier}\|$
STSM scheme	-	$(2 + \frac{2\log^3 n + 3\log^2 n - 11\log n}{6})\|E_{SHE}\|$	Same as User → Fog	$\|E_{SHE}\|$	$\|E_{SHE}\|$
PBTree-based scheme	-	$\Big((\log n - 1)(\log n + 2) + 2 * (\log n - 1)\Big)\|E_{SHE}\|$	Same as User → Fog	$\|E_{SHE}\|$	$\|E_{SHE}\|$

cessing schemes' performance. In our existing frameworks, we have considered just a single fog node at the fog layer to hand over encrypted queries from querying user to IoT devices and aggregate the IoT end-devices' responses. The layout can be extended into complex setups in which, for example, different numbers of fog nodes are deployed to improve the query response time and reduce the network load.

- Prototyping a real-world case and interactions between IoT end-devices and fog node by applying the proposed algorithms to evaluate and refine the prototypes in actual implementations.

- Applying our symmetric homomorphic scheme to other sectors. Since assuring security and privacy becomes the main concern in privacy-aware computation offloading, our light-weight symmetric homomorphic encryption can be applied differently within different disciplines and sectors. For example, symmetric homomorphic encryption can be employed in machine learning and image processing algorithms in which sensitive data may be illegitimately disclosed to others.

Bibliography

[1] Mohammad Aazam, Sherali Zeadally, and Khaled A Harras, *Fog computing architecture, evaluation, and future research directions*, IEEE Communications Magazine **56** (2018), no. 5, 46–52.

[2] Fizza Abbas, Rasheed Hussain, Junggab Son, and Heekuck Oh, *Privacy preserving cloud-based computing platform (ppccp) for using location based services*, 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, IEEE, 2013, pp. 60–66.

[3] Omar G Abood and Shawkat K Guirguis, *A survey on cryptography algorithms*, International Journal of Scientific and Research Publications **8** (2018), no. 7, 410–415.

[4] Nabil Abubaker, Leonard Dervishi, and Erman Ayday, *Privacy-preserving fog computing paradigm*, 2017 IEEE Conference on Communications and Network Security (CNS), IEEE, 2017, pp. 502–509.

[5] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti, *A survey on homomorphic encryption schemes: Theory and implementation*, ACM Computing Surveys (CSUR) **51** (2018), no. 4, 1–35.

[6] Zeyad Al-Odat and Samee Khan, *Constructions and attacks on hash functions*, 2019 International Conference on Computational Science and Computational Intelligence (CSCI), IEEE, 2019, pp. 139–144.

[7] Mobark Q Aldossari and Anna Sidorova, *Consumer acceptance of internet of things (iot): Smart home context*, Journal of Computer Information Systems **60** (2020), no. 6, 507–517.

[8] Roman Alexander, *Iota-introduction to the tangle technology: Everything you need to know about the revolutionary blockchain alternative*, Independently published, 2018.

[9] Arwa Alrawais, Abdulrahman Alhothaily, Chunqiang Hu, and Xiuzhen Cheng, *Fog computing for the internet of things: Security and privacy issues*, IEEE Internet Computing **21** (2017), no. 2, 34–42.

[10] Mohamed Ahzam Amanullah, Riyaz Ahamed Ariyaluran Habeeb, Fariza Hanum Nasaruddin, Abdullah Gani, Ejaz Ahmed, Abdul Salam Mohamed Nainar, Nazihah Md Akim, and Muhammad Imran, *Deep learning and big data technologies for iot security*, Computer Communications **151** (2020), 495–517.

[11] Wesam Al Amiri, Mohamed Baza, Karim Banawan, Mohamed Mahmoud, Waleed Alasmary, and Kemal Akkaya, *Privacy-preserving smart parking system using blockchain and private information retrieval*, arXiv preprint arXiv:1904.09703 (2019).

[12] Richardson Andrade and Nelson Rosa, *Midsecthings: Assurance solution for security smart homes in iot*, 2019 IEEE 19th International Symposium on High Assurance Systems Engineering (HASE), IEEE, 2019, pp. 171–178.

[13] Noah Noah Apthorpe, *Network privacy and user protection in the internet of things*, Ph.D. thesis, Princeton University, 2020.

[14] Haiyong Bao and Le Chen, *A lightweight privacy-preserving scheme with data*

integrity for smart grid communications, Concurrency and Computation: Practice and Experience **28** (2016), no. 4, 1094–1110.

[15] Haiyong Bao and Rongxing Lu, *A new differentially private data aggregation with fault tolerance for smart grid communications*, IEEE Internet of Things Journal **2** (2015), no. 3, 248–258.

[16] ———, *A lightweight data aggregation scheme achieving privacy preservation and data integrity with differential privacy and fault tolerance*, Peer-to-Peer Networking and Applications **10** (2017), no. 1, 106–121.

[17] Haiyong Bao, Rongxing Lu, Beibei Li, and Ruilong Deng, *Blithe: Behavior rule-based insider threat detection for smart grid*, IEEE Internet of Things Journal **3** (2015), no. 2, 190–205.

[18] Yaser Baseri, Abdelhakim Hafid, and Soumaya Cherkaoui, *Privacy preserving fine-grained location-based access control for mobile cloud*, computers & security **73** (2018), 249–265.

[19] Mohamed Baza, Noureddine Lasla, Mohamed Mahmoud, Gautam Srivastava, and Mohamed Abdallah, *B-ride: Ride sharing with privacy-preservation, trust and fair payment atop public blockchain*, IEEE Transactions on Network Science and Engineering (2019).

[20] Burton H Bloom, *Space/time trade-offs in hash coding with allowable errors*, Communications of the ACM **13** (1970), no. 7, 422–426.

[21] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim, *Evaluating 2-dnf formulas on ciphertexts*, Theory of Cryptography Conference, Springer, 2005, pp. 325–341.

[22] Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu, *Fog computing: A platform for internet of things and analytics*, Big data and internet of things: A roadmap for smart environments, Springer, 2014, pp. 169–186.

[23] Joppe W Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig, *Improved security for a ring-based fully homomorphic encryption scheme*, IMA International Conference on Cryptography and Coding, Springer, 2013, pp. 45–64.

[24] Rihab Boussada, Balkis Hamdane, Mohamed Elhoucine Elhdhili, and Leila Azouz Saidane, *Privacy-preserving aware data transmission for iot-based e-health*, Computer Networks **162** (2019), 106866.

[25] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan, *(leveled) fully homomorphic encryption without bootstrapping*, ACM Transactions on Computation Theory (TOCT) **6** (2014), no. 3, 13.

[26] Emmanuel Bresson, Dario Catalano, and David Pointcheval, *A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications*, International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2003, pp. 37–54.

[27] Christopher Brewster, Ioanna Roussaki, Nikos Kalatzis, Kevin Doolin, and Keith Ellis, *Iot in agriculture: Designing a europe-wide large-scale pilot*, IEEE communications magazine **55** (2017), no. 9, 26–33.

[28] Shariq Aziz Butt, Jorge Luis Diaz-Martinez, Tauseef Jamal, Arshad Ali, Emiro De-La-Hoz-Franco, and Muhammad Shoaib, *Iot smart health security threats*, 2019 19th International Conference on Computational Science and Its Applications (ICCSA), IEEE, 2019, pp. 26–31.

[29] Claude Castelluccia, Einar Mykletun, and Gene Tsudik, *Efficient aggregation of encrypted data in wireless sensor networks*, The second annual international conference on mobile and ubiquitous systems: networking and services, IEEE, 2005, pp. 109–117.

[30] Fei Chen and Alex X Liu, *Safeq: Secure and efficient query processing in sensor networks*, 2010 Proceedings IEEE INFOCOM, IEEE, 2010, pp. 1–9.

[31] Le Chen, Rongxing Lu, Zhenfu Cao, Khalid AlHarbi, and Xiaodong Lin, *Muda: Multifunctional data aggregation in privacy-preserving smart grid communications*, Peer-to-peer networking and applications **8** (2015), no. 5, 777–792.

[32] Peng Chen, Weini Zeng, Yong Zhu, and Yingyan Gu, *Secure range query based on privacy-preserving function in two-tiered sensor networks*, 2016 International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC), IEEE, 2016, pp. 1–6.

[33] Songlin Chen, Hong Wen, Jinsong Wu, Wenxin Lei, Wenjing Hou, Wenjie Liu, Aidong Xu, and Yixin Jiang, *Internet of things based smart grids supported by intelligent edge computing*, IEEE Access **7** (2019), 74089–74102.

[34] Yan-Da Chen, Muhammad Zulfan Azhari, and Jenq-Shiou Leu, *Design and implementation of a power consumption management system for smart home over fog-cloud computing*, 2018 3rd International Conference on Intelligent Green Building and Smart Grid (IGBSG), IEEE, 2018, pp. 1–5.

[35] Mung Chiang and Tao Zhang, *Fog and iot: An overview of research opportunities*, IEEE Internet of Things Journal **3** (2016), no. 6, 854–864.

[36] Fog Computing, *the internet of things: Extend the cloud to where the things are*, Cisco White Paper (2015).

[37] Mauro Conti, Ali Dehghantanha, Katrin Franke, and Steve Watson, *Internet of things security and forensics: Challenges and opportunities*, 2018.

[38] Hua Dai, Qingqun Ye, Xun Yi, Ruiliang He, Geng Yang, and Jinji Pan, *Vp2rq: Efficient verifiable privacy-preserving range query processing in two-tiered wire-*

less sensor networks, International Journal of Distributed Sensor Networks **12** (2016), no. 11, 1550147716675627.

[39] Amir Vahid Dastjerdi, Harshit Gupta, Rodrigo N Calheiros, Soumya K Ghosh, and Rajkumar Buyya, *Fog computing: Principles, architectures, and applications*, Internet of things, Elsevier, 2016, pp. 61–75.

[40] Parul Datta and Bhisham Sharma, *A survey on iot architectures, protocols, security and smart city based applications*, 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), IEEE, 2017, pp. 1–5.

[41] Lei Dong, Xuan Chen, Jianxiang Zhu, Hong Chen, Ke Wang, and Cuiping Li, *A secure collusion-aware and probability-aware range query processing in tiered sensor networks*, 2015 IEEE 34th Symposium on Reliable Distributed Systems (SRDS), IEEE, 2015, pp. 110–119.

[42] Joy Dutta and Sarbani Roy, *Iot-fog-cloud based architecture for smart city: Prototype of a smart building*, 2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence, IEEE, 2017, pp. 237–242.

[43] EETimes, *5 ways the food industry can improve food safety with the iot*, https://iot.eetimes.com/5-ways-the-food-industry-can-improve-food-safety-with-the-iot/, 2019.

[44] Binara NB Ekanayake, Malka N Halgamuge, and Ali Syed, *Security and privacy issues of fog computing for the internet of things (iot)*, Cognitive Computing for Big Data Systems Over IoT, Springer, 2018, pp. 139–174.

[45] Taher ElGamal, *A public key cryptosystem and a signature scheme based on*

discrete logarithms, IEEE transactions on information theory **31** (1985), no. 4, 469–472.

[46] Sarah M Erfani, Shanika Karunasekera, Christopher Leckie, and Udaya Parampalli, *Privacy-preserving data aggregation in participatory sensing networks*, 2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing, IEEE, 2013, pp. 165–170.

[47] Junfeng Fan and Frederik Vercauteren, *Somewhat practical fully homomorphic encryption.*, IACR Cryptology ePrint Archive **2012** (2012), 144.

[48] Kai Fan, Huiyue Xu, Longxiang Gao, Hui Li, and Yintang Yang, *Efficient and privacy preserving access control scheme for fog-enabled iot*, Future Generation Computer Systems **99** (2019), 134–142.

[49] Mohamed Amine Ferrag, Abdelouahid Derhab, Leandros Maglaras, Mithun Mukherjee, and Helge Janicke, *Privacy-preserving schemes for fog-based iot applications: Threat models, solutions, and challenges*, 2018 International Conference on Smart Communications in Network Technologies (SaCoNeT), IEEE, 2018, pp. 37–42.

[50] Keke Gai, Kim-Kwang Raymond Choo, Meikang Qiu, and Liehuang Zhu, *Privacy-preserving content-oriented wireless communication in internet-of-things*, IEEE Internet of Things Journal **5** (2018), no. 4, 3059–3067.

[51] Craig Gentry and Dan Boneh, *A fully homomorphic encryption scheme*, vol. 20, Stanford University Stanford, 2009.

[52] Craig Gentry, Z Brakerski, and V Vaikuntanathan, *Fully homomorphic encryption without bootstrapping*, Security **111** (2011), no. 111, 1–12.

[53] Jingjing Gu, Ruicong Huang, Li Jiang, Gongzhe Qiao, Xiaojiang Du, and Mohsen Guizani, *A fog computing solution for context-based privacy leakage detection for android healthcare devices*, Sensors **19** (2019), no. 5, 1184.

[54] Zhitao Guan, Yue Zhang, Longfei Wu, Jun Wu, Jing Li, Yinglong Ma, and Jingjing Hu, *Appa: An anonymous and privacy preserving data aggregation scheme for fog-enhanced iot*, Journal of Network and Computer Applications **125** (2019), 82–92.

[55] Amelie Gyrard, Antoine Zimmermann, and Amit Sheth, *Building iot-based applications for smart cities: How can ontology catalogs help?*, IEEE Internet of Things Journal **5** (2018), no. 5, 3978–3990.

[56] Riyaz Ahamed Ariyaluran Habeeb, Fariza Nasaruddin, Abdullah Gani, Ibrahim Abaker Targio Hashem, Ejaz Ahmed, and Muhammad Imran, *Real-time big data processing for anomaly detection: A survey*, International Journal of Information Management **45** (2019), 289–307.

[57] Vikas Hassija, Vinay Chamola, Vikas Saxena, Divyansh Jain, Pranav Goyal, and Biplab Sikdar, *A survey on iot security: Application areas, security threats, and solution architectures*, IEEE Access **7** (2019), 82721–82743.

[58] Amin Hosseinian-Far, Muthu Ramachandran, and Charlotte Lilly Slack, *Emerging trends in cloud computing, big data, fog computing, iot and smart living*, Technology for Smart Futures, Springer, 2018, pp. 29–40.

[59] Jianwei Hou, Leilei Qu, and Wenchang Shi, *A survey on internet of things security from data perspectives*, Computer Networks **148** (2019), 295–306.

[60] Cheng Huang, Dongxiao Liu, Jianbing Ni, Rongxing Lu, and Xuemin Shen, *Reliable and privacy-preserving selective data aggregation for fog-based iot*, 2018

IEEE International Conference on Communications (ICC), IEEE, 2018, pp. 1–6.

[61] Frank T Johnsen, Zbigniew Zieliński, Konrad Wrona, Niranjan Suri, Christoph Fuchs, Manas Pradhan, Janusz Furtak, Bogdan Vasilache, Vincenzo Pellegrini, Michał Dyk, et al., *Application of iot in military operations in a smart city*, 2018 International Conference on Military Communications and Information Systems (ICMCIS), IEEE, 2018, pp. 1–8.

[62] Ali Kanso, Hamdi Yahyaoui, and Mohammed Almulla, *Keyed hash function based on a chaotic map*, Information Sciences **186** (2012), no. 1, 249–264.

[63] Aykut Kanyilmaz and Aydin Cetin, *Fog based architecture design for iot with private nodes: A smart home application*, 2019 7th ICSG, IEEE, 2019, pp. 194–198.

[64] Ravneet Kaur and Amandeep Kaur, *Digital signature*, 2012 International Conference on Computing Sciences, IEEE, 2012, pp. 295–301.

[65] Tran Anh Khoa, Le Mai Bao Nhu, Hoang Hai Son, Nguyen Minh Trong, Cao Hoang Phuc, Nguyen Thi Hoang Phuong, Nguyen Van Dung, Nguyen Hoang Nam, Dong Si Thien Chau, and Dang Ngoc Minh Duc, *Designing efficient smart home management with iot smart lighting: A case study*, Wireless Communications and Mobile Computing **2020** (2020).

[66] Qinglei Kong, Rongxing Lu, Maode Ma, and Haiyong Bao, *Achieve location privacy-preserving range query in vehicular sensing*, Sensors **17** (2017), no. 8, 1829.

[67] ———, *A privacy-preserving and verifiable querying scheme in vehicular fog data dissemination*, IEEE Transactions on Vehicular Technology **68** (2018), no. 2, 1877–1887.

[68] Qinglei Kong, Rongxing Lu, Maode Ma, and Haiyong Bao, *A privacy-preserving and verifiable querying scheme in vehicular fog data dissemination*, IEEE Trans. Vehicular Technology **68** (2019), no. 2, 1877–1887.

[69] Qinglei Kong, Rongxing Lu, Maode Ma, and Haiyong Bao, *A privacy-preserving sensory data sharing scheme in internet of vehicles*, Future Generation Computer Systems **92** (2019), 644–655.

[70] Pankaj Kumar and Lokesh Chouhan, *A secure authentication scheme for iot application in smart home*, Peer-to-Peer Networking and Applications (2020), 1–19.

[71] Yongxuan Lai, Yifan Xu, Fan Yang, Wei Lu, and Quan Yu, *Privacy-aware query processing in vehicular ad-hoc networks*, Ad Hoc Networks (2019), 101876.

[72] Kevin Lewi and David J Wu, *Order-revealing encryption: New constructions, applications, and lower bounds*, Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 1167–1178.

[73] Guilin Li, Longjiang Guo, Xing Gao, and Minghong Liao, *Bloom filter based processing algorithms for the multi-dimensional event query in wireless sensor networks*, Journal of Network and Computer Applications **37** (2014), 323–333.

[74] Meng Li, Fan Wu, Guihai Chen, Liehuang Zhu, and Zijian Zhang, *How to protect query and report privacy without sacrificing service quality in participatory sensing*, 2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC), IEEE, 2015, pp. 1–7.

[75] Meng Li, Liehuang Zhu, and Xiaodong Lin, *Efficient and privacy-preserving carpooling using blockchain-assisted vehicular fog computing*, IEEE Internet of Things Journal **6** (2018), no. 3, 4573–4584.

[76] Meng Li, Liehuang Zhu, Zijian Zhang, Xiaojiang Du, and Mohsen Guizani, *Pros: a privacy-preserving route-sharing service via vehicular fog computing*, IEEE Access **6** (2018), 66188–66197.

[77] Xu Li, Rongxing Lu, Xiaohui Liang, Xuemin Shen, Jiming Chen, and Xiaodong Lin, *Smart community: an internet of things application*, IEEE Communications magazine **49** (2011), no. 11, 68–75.

[78] Haoran Liang, Jun Wu, Zhi Liu, and Jianhua Li, *Shape-unconstrained privacy-preserving range query for fog computing supported vehicular networks using image*, 2019 IEEE/CIC International Conference on Communications in China (ICCC), IEEE, 2019, pp. 683–688.

[79] Jinwen Liang, Zheng Qin, Sheng Xiao, Jixin Zhang, Hui Yin, and Keqin Li, *Privacy-preserving range query over multi-source electronic health records in public clouds*, Journal of Parallel and Distributed Computing (2019).

[80] Shushu Liu, An Liu, Zheng Yan, and Wei Feng, *Efficient lbs queries with mutual privacy preservation in iov*, Vehicular Communications **16** (2019), 62–71.

[81] Yi Liu, Weili Han, Yin Zhang, Lulu Li, Junyu Wang, and Lirong Zheng, *An internet-of-things solution for food safety and quality control: A pilot project in china*, Journal of Industrial Information Integration **3** (2016), 1–7.

[82] Javier Lopez, Ruben Rios, Feng Bao, and Guilin Wang, *Evolving privacy: From sensors to the internet of things*, Future Generation Computer Systems **75** (2017), 46–57.

[83] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan, *On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption*, Proceedings of the forty-fourth annual ACM symposium on Theory of computing, ACM, 2012, pp. 1219–1234.

[84] Rongxing Lu, *Privacy-enhancing aggregation techniques for smart grid communications*, Springer, 2016.

[85] ———, *A new communication-efficient privacy-preserving range query scheme in fog-enhanced iot*, IEEE Internet of Things Journal **6** (2018), no. 2, 2497–2505.

[86] Rongxing Lu, Kevin Heung, Arash Habibi Lashkari, and Ali A Ghorbani, *A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced iot*, IEEE Access **5** (2017), 3302–3312.

[87] Rongxing Lu, Xiaohui Liang, Xu Li, Xiaodong Lin, and Xuemin Shen, *Eppa: An efficient and privacy-preserving aggregation scheme for secure smart grid communications*, IEEE Transactions on Parallel and Distributed Systems **23** (2012), no. 9, 1621–1631.

[88] Hassan Mahdikhani and Rongxing Lu, *Achieving privacy-preserving multi dot-product query in fog computing-enhanced iot*, 2017 IEEE Global Communications Conference, GLOBECOM 2017, Singapore, December 4-8, 2017, IEEE, 2017, pp. 1–6.

[89] Hassan Mahdikhani, Rongxing Lu, Yandong Zheng, Jun Shao, and Ali A. Ghorbani, *Achieving $o(log^3 n)$ communication-efficient privacy-preserving range query in fog-based iot*, IEEE Internet Things J. **7** (2020), no. 6, 5220–5232.

[90] Hassan Mahdikhani, Samaneh Mahdavifar, Rongxing Lu, Hui Zhu, and Ali A. Ghorbani, *Achieving privacy-preserving subset aggregation in fog-enhanced iot*, IEEE Access **7** (2019), 184438–184447.

[91] Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya, *Fog com-*

puting: A taxonomy, survey and future directions, Internet of everything, Springer, 2018, pp. 103–130.

[92] Jeswanth Maseedu and Syam Kumar Pasupuleti, *Attribute based range search over encrypted data for privacy preserving in cloud computing*, 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, 2018, pp. 323–329.

[93] Alfred J Menezes, Jonathan Katz, Paul C Van Oorschot, and Scott A Vanstone, *Handbook of applied cryptography*, CRC press, 1996.

[94] Hichem Mrabet, Sana Belguith, Adeeb Alhomoud, and Abderrazak Jemai, *A survey of iot security based on a layered architecture of sensing and data analysis*, Sensors **20** (2020), no. 13, 3625.

[95] Ghulam Muhammad, SK Md Mizanur Rahman, Abdulhameed Alelaiwi, and Atif Alamri, *Smart health solution integrating iot and cloud: A case study of voice pathology monitoring*, IEEE Communications Magazine **55** (2017), no. 1, 69–73.

[96] Mithun Mukherjee, Rakesh Matam, Lei Shu, Leandros Maglaras, Mohamed Amine Ferrag, Nikumani Choudhury, and Vikas Kumar, *Security and privacy in fog computing: Challenges*, IEEE Access **5** (2017), 19293–19304.

[97] Chandra Sukanya Nandyala and Haeng-Kon Kim, *From cloud to fog and iot-based real-time u-healthcare monitoring for smart homes and hospitals*, International Journal of Smart Home **10** (2016), no. 2, 187–196.

[98] Jianbing Ni, Kuan Zhang, Xiaodong Lin, and Xuemin Sherman Shen, *Securing fog computing for internet of things applications: Challenges and solutions*, IEEE Communications Surveys & Tutorials **20** (2017), no. 1, 601–628.

[99] Boubakr Nour, Kashif Sharif, Fan Li, and Yu Wang, *Security and privacy challenges in information-centric wireless internet of things networks*, IEEE Security & Privacy **18** (2019), no. 2, 35–45.

[100] Oscar Novo, *Blockchain meets iot: An architecture for scalable access management in iot*, IEEE Internet of Things Journal **5** (2018), no. 2, 1184–1195.

[101] Lu Ou, Hui Yin, Zheng Qin, Sheng Xiao, Guangyi Yang, and Yupeng Hu, *An efficient and privacy-preserving multiuser cloud-based lbs query scheme*, Security and communication networks **2018** (2018).

[102] Mustafa Ozger, Oktay Cetinkaya, and Ozgur B Akan, *Energy harvesting cognitive radio networking for iot-enabled smart grid*, Mobile Networks and Applications **23** (2018), no. 4, 956–966.

[103] Ali Padyab and Anna Ståhlbröst, *Exploring the dimensions of individual privacy concerns in relation to the internet of things use situations*, Digital Policy, Regulation and Governance (2018).

[104] Pascal Paillier, *Public-key cryptosystems based on composite degree residuosity classes*, International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 1999, pp. 223–238.

[105] Parthasarathy Panchatcharam and S Vivekanandan, *Internet of things (iot) in healthcare–smart health and surveillance, architectures, security analysis and data transfer: a review*, International Journal of Software Innovation (IJSI) **7** (2019), no. 2, 21–40.

[106] Youyang Qu, Shui Yu, Wanlei Zhou, Sancheng Peng, Guojun Wang, and Ke Xiao, *Privacy of things: Emerging challenges and opportunities in wireless internet of things*, IEEE Wireless Communications **25** (2018), no. 6, 91–97.

[107] Abdul Rauf, Riaz Ahmed Shaikh, and Asadullah Shah, *Security and privacy for iot and fog computing paradigm*, 2018 15th Learning and Technology Conference (L&T), IEEE, 2018, pp. 96–101.

[108] Partha Pratim Ray, *A survey on internet of things architectures*, Journal of King Saud University-Computer and Information Sciences **30** (2018), no. 3, 291–319.

[109] Hao Ren, Hongwei Li, Hao Chen, Michael Kpiebaareh, and Lian Zhao, *Efficient privacy-preserving circular range search on outsourced spatial data*, 2016 IEEE International Conference on Communications (ICC), IEEE, 2016, pp. 1–7.

[110] Hao Ren, Hongwei Li, Yuanshun Dai, Kan Yang, and Xiaodong Lin, *Querying in internet of things with privacy preserving: Challenges, solutions and opportunities*, IEEE Network **32** (2018), no. 6, 144–151.

[111] Kumari Renuka, Satya Narayan Das, and K Hemant Reddy, *An efficient context management approach for iot*, IUP Journal of Information Technology **14** (2018), no. 2, 24–35.

[112] Ronald L Rivest, Adi Shamir, and Leonard Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM **21** (1978), no. 2, 120–126.

[113] ———, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM **21** (1978), no. 2, 120–126.

[114] Alisha Rohilla, Mehak Khurana, and Latika Singh, *Location privacy using homomorphic encryption over cloud*, International Journal of Computer Network and Information Security **11** (2017), no. 8, 32.

[115] Rahul Saha, Gulshan Kumar, Mritunjay Kumar Rai, Reji Thomas, and Se-Jung Lim, *Privacy ensured e-healthcare for fog-enhanced iot based applications*, IEEE Access **7** (2019), 44536–44543.

[116] Mohammad A Salahuddin, Ala Al-Fuqaha, Mohsen Guizani, Khaled Shuaib, and Farag Sallabi, *Softwarization of internet of things infrastructure for secure and smart healthcare*, arXiv preprint arXiv:1805.11011 (2018).

[117] Bharath K. Samanthula and Wei Jiang, *Efficient privacy-preserving range queries over encrypted data in cloud computing*, 2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, June 28 - July 3, 2013, 2013, pp. 51–58.

[118] Bharath Kumar Samanthula, James Liporace, and Martin Issiov, *A privacy-aware and user-centric approach for query processing in cloud*, 2018 IEEE International Conference on Smart Cloud (SmartCloud), IEEE, 2018, pp. 54–59.

[119] C Sanchez-Avila and R Sanchez-Reillol, *The rijndael block cipher (aes proposal): a comparison with des*, Proceedings IEEE 35th Annual 2001 International Carnahan Conference on Security Technology (Cat. No. 01CH37186), IEEE, 2001, pp. 229–234.

[120] Daniel AF Saraiva, Valderi Reis Quietinho Leithardt, Diandre de Paula, Andre Sales Mendes, Gabriel Villarrubia González, and Paul Crocker, *Prisec: Comparison of symmetric key algorithms for iot devices*, Sensors **19** (2019), no. 19, 4312.

[121] Kinza Sarwar, Sira Yongchareon, and Jian Yu, *A brief survey on iot privacy: Taxonomy, issues and future trends*, International Conference on Service-Oriented Computing, Springer, 2018, pp. 208–219.

[122] Neetesh Saxena, Bong Jun Choi, and Rongxing Lu, *Authentication and authorization scheme for various user roles and devices in smart grid*, IEEE Trans. Inf. Forensics Secur. **11** (2016), no. 5, 907–921.

[123] Pallavi Sethi and Smruti R Sarangi, *Internet of things: architectures, protocols, and applications*, Journal of Electrical and Computer Engineering **2017** (2017).

[124] Bilal Shabandri and Piyush Maheshwari, *Enhancing iot security and privacy using distributed ledgers with iota and the tangle*, 2019 6th International Conference on Signal Processing and Integrated Networks (SPIN), IEEE, 2019, pp. 1069–1075.

[125] Jun Shao, Rongxing Lu, and Xiaodong Lin, *Fine: A fine-grained privacy-preserving location-based service framework for mobile devices*, IEEE INFOCOM 2014-IEEE Conference on Computer Communications, IEEE, 2014, pp. 244–252.

[126] Suhaila Omer Sharif and SP Mansoor, *Performance analysis of stream and block cipher algorithms*, 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), vol. 1, IEEE, 2010, pp. V1–522.

[127] Simar Preet Singh, Anand Nayyar, Rajesh Kumar, and Anju Sharma, *Fog computing: from architecture to edge computing and big data processing*, The Journal of Supercomputing (2018), 1–36.

[128] Nigel P Smart and Frederik Vercauteren, *Fully homomorphic encryption with relatively small key and ciphertext sizes*, International Workshop on Public Key Cryptography, Springer, 2010, pp. 420–443.

[129] William Stallings, *Cryptography and network security: Principles and practice, 8/e*, Pearson Education India, 2020.

[130] TechTarget, *The impact of iot on big data*, https://internetofthingsagenda.techtarget.com/blog/IoT-Agenda/The-impact-of-IoT-on-big-data, 2019.

[131] Anselme Tueno and Florian Kerschbaum, *Secure range queries for multiple users*, arXiv preprint arXiv:1802.01138 (2018).

[132] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan, *Fully homomorphic encryption over the integers*, Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2010, pp. 24–43.

[133] Chaitanya Varma, *A study of the ecc, rsa and the diffie-hellman algorithms in network security*, 2018 International Conference on Current Trends towards Converging Technologies (ICCTCT), IEEE, 2018, pp. 1–4.

[134] Prabal Verma and Sandeep K Sood, *Fog assisted-iot enabled patient health monitoring in smart homes*, IEEE Internet of Things Journal **5** (2018), no. 3, 1789–1796.

[135] Alexandre Viejo and David Sánchez, *Secure and privacy-preserving orchestration and delivery of fog-enabled iot services*, Ad Hoc Networks **82** (2019), 113–125.

[136] Boyang Wang, Yantian Hou, Ming Li, Haitao Wang, and Hui Li, *Maple: scalable multi-dimensional range search over encrypted cloud data with tree-based index*, Proceedings of the 9th ACM symposium on Information, computer and communications security, ACM, 2014, pp. 111–122.

[137] Boyang Wang, Ming Li, Haitao Wang, and Hui Li, *Circular range search on encrypted spatial data*, 2015 IEEE Conference on Communications and Network Security (CNS), IEEE, 2015, pp. 182–190.

[138] Peng Wang and Chinya V. Ravishankar, *Secure and efficient range queries on outsourced databases using rp-trees*, 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013, 2013, pp. 314–325.

[139] Qian Wang, Meiqi He, Minxin Du, Sherman S. M. Chow, Russell W. F. Lai, and Qin Zou, *Searchable encryption over feature-rich data*, IEEE Transactions on Dependable and Secure Computing **15** (2018), no. 3, 496–510.

[140] Qian Wang, Jing Huang, Yanjiao Chen, Cong Wang, Fu Xiao, and Xiangyang Luo, *prost: Privacy-preserving and truthful online double auction for spectrum allocation*, IEEE Transactions on Information Forensics and Security **14** (2018), no. 2, 374–386.

[141] Tian Wang, Jiandian Zeng, Md Zakirul Alam Bhuiyan, Hui Tian, Yiqiao Cai, Yonghong Chen, and Bineng Zhong, *Trajectory privacy preservation based on a fog structure for cloud location services*, IEEE Access **5** (2017), 7692–7701.

[142] Mi Wen, Rongxing Lu, Kuan Zhang, Jingsheng Lei, Xiaohui Liang, and Xuemin Shen, *Parq: A privacy-preserving range query scheme over encrypted metering data for smart grid*, IEEE Transactions on Emerging Topics in Computing **1** (2013), no. 1, 178–191.

[143] Hao-Tian Wu, Yiu-ming Cheung, Zhiyuan Yang, and Shaohua Tang, *A high-capacity reversible data hiding method for homomorphic encrypted images*, Journal of Visual Communication and Image Representation **62** (2019), 87–96.

[144] Guowen Xu and Hongwei Li, *Achieving fine-grained multi-keyword ranked search over encrypted cloud data.*, IACR Cryptology ePrint Archive (2018).

[145] Mengmeng Yang, Tianqing Zhu, Bo Liu, Yang Xiang, and Wanlei Zhou, *Ma-*

chine learning differential privacy with multifunctional aggregation in a fog computing architecture, IEEE Access **6** (2018), 17119–17129.

[146] Xue Yang, Fan Yin, and Xiaohu Tang, *A fine-grained and privacy-preserving query scheme for fog computing-enhanced location-based service*, Sensors **17** (2017), no. 7, 1611.

[147] Yang Yang, Xiliang Luo, Xiaoli Chu, and Ming-Tuo Zhou, *Fog-enabled smart home and user behavior recognition*, Fog-Enabled Intelligent IoT Systems, Springer, 2020, pp. 185–210.

[148] Yuchen Yang, Longfei Wu, Guisheng Yin, Lijie Li, and Hongbin Zhao, *A survey on security and privacy issues in internet-of-things*, IEEE Internet of Things Journal **4** (2017), no. 5, 1250–1258.

[149] Muneer Bani Yassein, Shadi Aljawarneh, Ethar Qawasmeh, Wail Mardini, and Yaser Khamayseh, *Comprehensive study of symmetric key and asymmetric key encryption algorithms*, 2017 international conference on engineering and technology (ICET), IEEE, 2017, pp. 1–7.

[150] Abdulsalam Yassine, Shailendra Singh, M Shamim Hossain, and Ghulam Muhammad, *Iot big data analytics for smart homes with fog and cloud computing*, Future Generation Computer Systems **91** (2019), 563–573.

[151] Nafiseh Izadi Yekta and Rongxing Lu, *Pquery: Achieving privacy-preserving query with communication efficiency in internet of things*, 2017 IEEE 86th Vehicular Technology Conference (VTC-Fall), IEEE, 2017, pp. 1–5.

[152] ———, *Xrquery: Achieving communication-efficient privacy-preserving query for fog-enhanced iot*, 2018 IEEE International Conference on Communications (ICC), IEEE, 2018, pp. 1–6.

[153] Shanhe Yi, Zhengrui Qin, and Qun Li, *Security and privacy issues of fog computing: A survey*, International conference on wireless algorithms, systems, and applications, Springer, 2015, pp. 685–695.

[154] Xingliang Yuan, Xinyu Wang, Cong Wang, Baochun Li, Xiaohua Jia, et al., *Enabling encrypted rich queries in distributed key-value stores*, IEEE Transactions on Parallel and Distributed Systems **30** (2018), no. 6, 1283–1297.

[155] Juru Zeng, Lei Dong, Yao Wu, Hong Chen, Cuiping Li, and Shan Wang, *Privacy-preserving and multi-dimensional range query in two-tiered wireless sensor networks*, GLOBECOM 2017-2017 IEEE Global Communications Conference, IEEE, 2017, pp. 1–7.

[156] Juru Zeng, Yuncheng Wu, Yao Wu, Hong Chen, Cuiping Li, and Shan Wang, *Energy-efficient and privacy-preserving range query in participatory sensing*, 2016 IEEE Trustcom/BigDataSE/ISPA, IEEE, 2016, pp. 876–883.

[157] Xiaoying Zhang, Lei Dong, Hui Peng, Hong Chen, Deying Li, and Cuiping Li, *Achieving efficient and secure range query in two-tiered wireless sensor networks*, IEEE 22nd International Symposium of Quality of Service, IWQoS 2014, Hong Kong, China, May 26-27, 2014, 2014, pp. 380–388.

[158] Lei Zhao, Jiadai Wang, Jiajia Liu, and Nei Kato, *Optimal edge resource allocation in iot-based smart cities*, IEEE Network **33** (2019), no. 2, 30–35.

[159] Serena Zheng, Noah Apthorpe, Marshini Chetty, and Nick Feamster, *User perceptions of smart home iot privacy*, Proceedings of the ACM on Human-Computer Interaction **2** (2018), no. CSCW, 1–20.

[160] Yandong Zheng, Rongxing Lu, Yunguo Guan, Jun Shao, and Hui Zhu, *Efficient and privacy-preserving similarity range query over encrypted time series data*, IEEE Transactions on Dependable and Secure Computing (2021).

[161] Jun Zhou, Zhenfu Cao, Xiaolei Dong, and Athanasios V Vasilakos, *Security and privacy for cloud-based iot: Challenges*, IEEE Communications Magazine **55** (2017), no. 1, 26–33.

[162] Liming Zhou, Yingzi Shan, and Lu Chen, *Secure and efficient cluster-based range query processing in wireless sensor networks*, Journal of Electrical and Computer Engineering **2018** (2018).

[163] Hui Zhu, Rongxing Lu, Cheng Huang, Le Chen, and Hui Li, *An efficient privacy-preserving location-based services query scheme in outsourced cloud*, IEEE Transactions on Vehicular Technology **65** (2015), no. 9, 7729–7739.

[164] Xiaojie Zhu, Erman Ayday, and Roman Vitenberg, *A privacy-preserving framework for outsourcing location-based services to the cloud*, IEEE Transactions on Dependable and Secure Computing (2019).

Appendix A

This appendix provides the Java source and test code for the proposed symmetric homomorphic encryption (SHE) scheme.

Listing A.1: SHE Parameters

```
1  import java.math.BigInteger;
2  import java.util.List;
3
4  public class SHSParamters {
5      protected List<BigInteger> param;
6
7      public SHSParamters(List<BigInteger> param) {
8          this.param = param;
9      }
10
11     public List<BigInteger> getParams() {
12         return param;
13     }
14
15     public void setParams(List<BigInteger> key) {
16         this.param = key;
17     }
18 }
```

Listing A.2: SHE Source Code

```
1  import java.math.BigInteger;
2  import java.security.SecureRandom;
3  import java.util.ArrayList;
4  import java.util.Arrays;
5
6  //SHS = Symmetric Homomorphic Scheme
7  public final class SymHomSch {
8
9      private SecureRandom rnd = new SecureRandom();
10     private int k0, k1, k2;
11     private  SHSParamters secretParam; //sp
```

```
12      private  SHSParamters publicParam; //pp
13      public  void KeyGen(int param_k0, int param_k1, int param_k2)
14      {
15
16      k0 = param_k0; //1024; Length of large prime numbers p and q
17      k1 = param_k1; //30; Length of message and message space
18      k2 = param_k2;
19      //80; Length of parameter L and generated random values in encryption method
20
21  //k0=2048, k1=300, k2=400
22  //k1=300 can support 10^90 (log(10)/log2()= 3.322 and 300/3.322 = 90)
23  //Single multiplication needs 2k2+2k2=4*k2 and 4K2<<k0 4*400 => 1600 << 2048 (Correct!)
24  //Two multiplication needs 2k2+2k2+2k2=6*k2 and 6k2<<k0 ==> 6*400<< 2048 (Incorrect!)
25
26
27      BigInteger p = new BigInteger(k0, 40, rnd); // Certainty = 40
28      //p = BigInteger.probablePrime(k0, rnd);
29      BigInteger q = new BigInteger(k0, 40, rnd); // Certainty = 40
30      //q = BigInteger.probablePrime(k0, rnd);
31
32      BigInteger N = p.multiply(q);
33
34      //L = BigInteger.probablePrime(k2, rnd);
35      BigInteger L = new BigInteger(k2, rnd).add(BigInteger.ONE);
36          //L in {1,2,3,..., 2^k2}; e.g., k2=80;
37
38      secretParam = new SHSParamters(new ArrayList<>(Arrays.asList(p, L, N, q)));
39      publicParam = new SHSParamters(new ArrayList<>(Arrays.asList(N)));
40
41  }
42
43
44  public BigInteger Enc(BigInteger message, SHSParamters Param) {
45          //Param is a secret paramater. (p,L,N)
46      BigInteger p = Param.getParams().get(0);
47      BigInteger L = Param.getParams().get(1);
48      BigInteger N = Param.getParams().get(2);
49
50
51      BigInteger r = (new BigInteger(k2, rnd)).add(BigInteger.ONE);
52      //r in {1,2,3,..., 2^k2}; e.g., k2=80;
53      System.out.println("|r|=k2: " + r.bitLength());//+ " " +r);
54      BigInteger rp = ((new BigInteger(k0*2, rnd)).add(BigInteger.ONE)).mod(N);
55      //r' in ZN; |N|=|p|*|q|=k0*K0;
56      System.out.println("|r' in ZN|: " + rp.bitLength());//+ " " +rp);
57      return (((r.multiply(L)).add(message)).multiply((BigInteger.ONE).
58              add(rp.multiply(p)))).mod(N);
59  }
60
61
62   public BigInteger Dec(BigInteger cipher, SHSParamters Param) {
63
64      BigInteger p = Param.getParams().get(0);
65      BigInteger L = Param.getParams().get(1);
```

```
66      BigInteger N = Param.getParams().get(2);
67
68      System.out.println("cipher.BitLength: " + cipher.bitLength());
69      BigInteger x = cipher.mod(p);
70      BigInteger y;
71      System.out.println("x.BitLength: " + x.bitLength());
72      if (x.bitLength() < k1+15) // x = c mod p; x in Message space.
73      {// if it had been set to k1+15 it can recover m values grater than message space.
74          y = x.mod(L);
75          System.out.println("y.BitLength: " + y.bitLength());
76          System.out.println("Case -1");
77          return x.mod(L); // or return x;
78      }else if ((x.bitLength() > k2+5) &&  (x.bitLength() < k0-50))
79      // x = c mod p; |L| < |x| << |p|
80      {
81          y = x.mod(L);
82          System.out.println("y.BitLength: " + y.bitLength());
83          if (y.bitLength()<k1+15)
84         // if it had been set to k1+15 it can recover m values grater than message space.
85          {
86              System.out.println("Case 1");
87              return y;
88          }
89          else if ((y.bitLength()<k2+10)&&(y.bitLength()>k2-10)) // y close to L
90          {
91              System.out.println("Case 2.1");
92              return y.subtract(L);
93          }
94          else
95          {
96              System.out.println("Case unknown01 - due to bad boundary setting.");
97              return BigInteger.valueOf((long)-1);
98          }
99      }else if((x.bitLength()<k0+5 && x.bitLength()>k0-5))// x = c mod p; x close to p
100     {
101         y = x.mod(L);
102         System.out.println("y.BitLength: " + y.bitLength());
103         if ( y.compareTo(p.mod(L))>=0) // y>= p mod L
104         {
105             System.out.println("Case 3");
106             return y.subtract(p.mod(L));
107         }
108         else if(y.compareTo(p.mod(L))==-1)// y < p mod L
109         {
110             System.out.println("Case 4 or 2.2");
111             return y.subtract(p.mod(L));
112         }
113         else
114         {
115             System.out.println("Case unknown02");
116             return BigInteger.valueOf((long)-1);
117         }
118     }
119     else
```

```
120     {
121         System.out.println("Case unknown03");
122         return BigInteger.valueOf((long)-1);
123     }
124    }
125
126  public SHSParamters getPublicParams() {
127        return publicParam;
128  }
129
130  public SHSParamters getSecretParams() {
131         return secretParam;
132  }
133 // Homomorphic Add (Cipher1, Chipher2)
134  public static BigInteger Add(BigInteger cipher1, BigInteger cipher2, SHSParamters Param){
135     //Param is a public parameter(N)
136     return cipher1.add(cipher2).mod(Param.getParams().get(0));
137     }
138 // Homomorphic Multiplication (Cipher1, Chipher2)
139  public static BigInteger Mul(BigInteger cipher1, BigInteger cipher2, SHSParamters Param){
140     //Param is a public parameter(N)
141     return cipher1.multiply(cipher2).mod(Param.getParams().get(0));
142 }
143 // Homomorphic Subtract (Cipher1, Chipher2)
144  public static BigInteger Sub(BigInteger cipher1, BigInteger cipher2, SHSParamters Param){
145     //Param is a public parameter(N)
146     return cipher1.subtract(cipher2).mod(Param.getParams().get(0));
147     }
148 }
```

Listing A.3: SHE Test

```
1  import java.math.BigDecimal;
2  import java.math.BigInteger;
3  import java.security.SecureRandom;
4  import java.util.ArrayList;
5  import java.util.Arrays;
6
7  import javax.sound.midi.SysexMessage;
8
9  public class SymHOMSchTest {
10
11     public static void main(String[] args) {
12
13     SymHomSch shs =  new SymHomSch();
14     shs.KeyGen(2300, 35, 45);
15
16 //4*k2<<k0 to support one multiplication of enc values, enc(a)*enc(b)
17 //2k2+2K2 = 4*k2 for one multiplication enc(a)*enc(b)
18 //2k2+2k2+2k2 = 6*k2 to support two multiplications, enc(a)*enc(b)*enc(c)
19 //and 6*k2 = 6*400 << 3500 (k0)
20 //2k2+2k2+2k2+2k2 = 8*k2 to support three multiplication, enc(a)*enc(b)*enc(c)*enc(d)
21 // and 8*k2= 8*400 << 3500
22 //To support w multiplication, (w+1)*2K2 << k0 should be satisfied.
```

```
23
24      SHSParamters pp = shs.getPublicParams();
25      SHSParamters sp = shs.getSecretParams();
26
27
28      System.out.println("N="+pp.getParams().get(0).bitLength()+" "+pp.getParams().get(0));
29      System.out.println("P="+sp.getParams().get(0).bitLength()+" "+sp.getParams().get(0));
30      System.out.println("Q="+sp.getParams().get(3).bitLength()+" "+sp.getParams().get(3));
31      System.out.println("N="+sp.getParams().get(2).bitLength()+" "+sp.getParams().get(2));
32      System.out.println("L="+sp.getParams().get(1).bitLength()+" "+sp.getParams().get(1));
33
34      int[] plainText = new int[]{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1};
35      BigInteger[] cipherText = new BigInteger[plainText.length];
36
37      for(int i=0; i<plainText.length; i++){
38         cipherText[i] = shs.Enc(BigInteger.valueOf(plainText[i]), sp);
39         System.out.println("Dec("+i+"): "+ shs.Dec(cipherText[i], sp));
40      }
41
42      BigInteger cipherMulResult = BigInteger.ONE;
43      BigInteger cipherAddResult = BigInteger.ZERO;
44      //BigInteger cipherResult = cipherText[0];
45      for(int i=0; i<plainText.length; i++){
46         cipherMulResult = shs.Mul(cipherMulResult, cipherText[i], pp);
47         cipherAddResult = shs.Add(cipherAddResult, cipherText[i], pp);
48         System.out.println("Dec_Mul("+i+"): "+ shs.Dec(cipherMulResult, sp));
49         System.out.println("Dec_ADD("+i+"): "+ shs.Dec(cipherAddResult, sp));
50         System.out.println("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%");
51      }
52
53      //System.exit(0);
54
55      BigInteger s = shs.Enc(new BigInteger("1000"), sp);
56      BigInteger t = shs.Enc(new BigInteger("2000"), sp);
57      BigInteger res = new BigInteger("1");
58
59      System.out.println(shs.Dec(s, sp));
60      res = s.subtract(new BigInteger("3"));
61      System.out.println(shs.Dec(res, sp));
62      res = shs.Sub(s, t, shs.getPublicParams());
63      System.out.println(shs.Dec(res ,sp));
64
65      //System.exit(0);
66
67
68      BigInteger x = shs.Enc(new BigInteger("30"), sp);
69      System.out.println(shs.Dec(x, sp));
70      System.out.println("===== Enc(a) op plaintext =====");
71      System.out.println(shs.Dec(x.add(new BigInteger("20")), sp));
72      System.out.println(shs.Dec(x.add(new BigInteger("-20")), sp));
73      System.out.println(shs.Dec(x.subtract(new BigInteger("20")), sp));
74      System.out.println(shs.Dec(x.subtract(new BigInteger("-20")), sp));
75      System.out.println(shs.Dec(x.multiply(new BigInteger("20")), sp));
76      System.out.println(shs.Dec(x.multiply(new BigInteger("-20")), sp));
```

```
77      System.out.println("===== Enc(a) op Enc(b) =====");
78      BigInteger y = shs.Enc(new BigInteger("10"), sp);
79      System.out.println(shs.Dec(y, sp));
80      System.out.println(shs.Dec(x.add(y), sp));
81      System.out.println(shs.Dec(x.subtract(y), sp));
82      System.out.println(shs.Dec(x.multiply(y), sp));//cipher.BitLength=8188
83      System.out.println(shs.Dec(shs.Mul(x,y,pp), sp));//cipher.BitLength=4095
84      System.out.println("===== Enc(a) op Enc(-b) =====");
85      BigInteger z = shs.Enc(new BigInteger("-10"), sp);
86      System.out.println(shs.Dec(z, sp));
87      System.out.println(shs.Dec(x.add(z), sp));
88      System.out.println(shs.Dec(x.subtract(z), sp));
89      System.out.println(shs.Dec(x.multiply(z), sp));//cipher.BitLength=8190
90      System.out.println(shs.Dec(shs.Mul(x,z,pp), sp));//cipher.BitLength=4096
91
92      BigInteger a = shs.Enc(new BigInteger("-23"), sp);
93      System.out.println(shs.Dec(a, sp));
94      System.out.println("===== Enc(-a) op plaintext =====");
95      System.out.println(shs.Dec(a.add(new BigInteger("20")), sp));
96      System.out.println(shs.Dec(a.add(new BigInteger("-20")), sp));
97      System.out.println(shs.Dec(a.subtract(new BigInteger("20")), sp));
98      System.out.println(shs.Dec(a.subtract(new BigInteger("-20")), sp));
99      System.out.println(shs.Dec(a.multiply(new BigInteger("20")), sp));
100     System.out.println(shs.Dec(a.multiply(new BigInteger("-20")), sp));
101     System.out.println("===== Enc(-a) op Enc(b) =====");
102     BigInteger b = shs.Enc(new BigInteger("1000"), sp);
103     System.out.println(shs.Dec(b, sp));
104     System.out.println(shs.Dec(a.add(b), sp));
105     System.out.println(shs.Dec(a.subtract(b), sp));
106     System.out.println(shs.Dec(a.multiply(b), sp));//cipher.BitLength=8188
107     System.out.println(shs.Dec(shs.Mul(a,b,pp), sp));//cipher.BitLength=4095
108     System.out.println("===== Enc(-a) op Enc(-b) =====");
109     BigInteger c = shs.Enc(new BigInteger("-1000"), sp);
110     System.out.println(shs.Dec(c, sp));
111     System.out.println(shs.Dec(a.add(c), sp));
112     System.out.println(shs.Dec(a.subtract(c), sp));
113     System.out.println(shs.Dec(a.multiply(c), sp));//cipher.BitLength=8192
114     System.out.println(shs.Dec(shs.Mul(a,c,pp), sp));//cipher.BitLength=4095
115
116     System.out.println("== Enc(b) * Enc(b) ==4k2 {<<} k0==");
117     System.out.println(shs.Dec(shs.Mul(b,b,pp), sp));
118 //with k0=2048 and k2=400 (4*400<<2048) It can be processed.
119     System.out.println("== Enc(b) * Enc(b) * Enc(b) ==6k2 {<<}
120     k0==SET K0=2500 or 2600 for  k2=400 ==");
121     System.out.println(shs.Dec(shs.Mul(b,shs.Mul(b,b,pp),pp), sp));
122 //with k0=2048 and k2=400 (6*400 IS NOT <<2048) It can NOT be processed.
123     System.out.println("== Enc(b) * Enc(b) * Enc(b) * Enc(b)==8k2 {<<}
124     k0==SET k0=3300 or 3400 for k2=400 ==");
125     System.out.println(shs.Dec(shs.Mul(b,shs.Mul(b,shs.Mul(b,b,pp),pp),pp), sp));
126 //with k0=2048 and k2=400 (8*400 IS NOT <<2048) It can NOT be processed.
127     System.out.println("== Enc(b) * Enc(b) * Enc(b) * Enc(b) * 5 ==8k2 {<<}
128     k0==SET k0=3300 or 3400 for k2=400 ==");
129     System.out.println(shs.Dec(shs.Mul(b,shs.Mul(b,shs.Mul(b,b,pp),pp),pp).
130     multiply(BigInteger.valueOf(5)), sp));
```

```
131 //with k0=2048 and k2=400 (8*400 IS NOT <<2048) It can NOT be processed.
132     }
133 }
```

Vita

Candidate's full name: **Hassan Mahdikhani**

University attended (with dates and degrees obtained):

University of New Brunswick, Fredericton, Canada, 2017-2021, PhD
Iran University of Science and Technology, Tehran, Iran, 2003-2006, MEng
Kharazmi University, Tehran, Iran, 1997-2001, BEng

Journal Publications:

[J1] **Hassan Mahdikhani**, Rongxing Lu, Jun Shao, and Ali Ghorbani, Using Reduced Paths to Achieve Efficient Privacy-Preserving Range Query in Fog-Based IoT, IEEE Internet of Things Journal, 2020.

[J2] **Hassan Mahdikhani**, Rongxing Lu, Yandong Zheng, Jun Shao, and Ali Ghorbani, Achieving $O(\log^3 n)$ Communication-Efficient Privacy-Preserving Range Query in Fog-based IoT, IEEE Internet of Things Journal (2020), no. 6, pp. 5220-5232.

[J3] **Hassan Mahdikhani**, Rasoul Shahsavarifar, Rongxing Lu, and David Bremner, Achieve Privacy-Preserving Simplicial Depth Query over Collaborative Cloud Servers, Peer-to-Peer Networking and Applications 13 (2020), no. 1, 412-423.

[J4] **Hassan Mahdikhani**, Samaneh Mahdavifar, Rongxing Lu, Hui Zhu, and Ali Ghorbani, Achieving Privacy-Preserving Subset Aggregation in Fog-Enhanced IoT, IEEE Access, vol. 7, pp. 184438–184447, 2019.

Conference Publications:

[C1] **Hassan Mahdikhani**, Rongxing Lu, Yandong Zheng, and Ali Ghorbani, Achieving Efficient and Privacy-Preserving Range Query in Fog-Enhanced IoT with Bloom Filter, IEEE International Conference on Communications (ICC), ICC 2020, IEEE,

pp.1-6.

[C2] Shlomi Linoy, **Hassan Mahdikhani**, Suprio Ray, Rongxing Lu, Natalia Stakhanova, and Ali Ghorbani, Scalable Privacy-Preserving Query Processing over Ethereum Blockchain, 2019 IEEE International Conference on Blockchain (Blockchain), IEEE, 2019, pp. 398–404.

[C3] **Hassan Mahdikhani** and Rongxing Lu, Achieving Privacy-Preserving Multi Dot-Product Query in Fog Computing-Enhanced IoT, 2017 IEEE Global Communications Conference, GLOBECOM 2017, Singapore, December 4-8, 2017, IEEE, 2017, pp. 1-6.