Privacy-Preserving Data Aggregation in a Peer-to-Peer Network

A Multiparty Computation Approach

by

Prahesa Kusuma Setia

in partial fulfillment of the requirements for the degree of

Master of Science
in Computer Science

at the Delft University of Technology,
defended on 29 November 2017 at 10:00 AM.

Supervisor:	Dr. Zekeriya Erkin	
Thesis committee:	Prof. Dr. Dick H.J Epema,	TU Delft
	Dr. Klaus Hildebrandt,	TU Delft

Abstract

Current interconnected society provides us with numerous devices communicating with one another. Exchange of data thus become an integral part in our live. Data become valuable commodity in today's setting because of their usage by individual and other interested parties. Several parties may be interested in computing a function over their data while still want to keep the information on their own data private.

Prior research on computing function in privacy preserving way in the domain of smart power-grid, e-metering system, wireless sensor network, and smart phone sensing generally focus on their own application and assume a total control and the static structure of the network. Moreover, a new paradigm in the field of decentralized power-grid requires privacy preserving solution to be applicable without existence of central authority. We propose two privacy preserving data aggregation protocols in peer-to-peer network scenario where there is such central authority involved. The first protocol utilizes additive homomorphism properties of Pailier scheme and the second protocol utilizes secret sharing. Both of the protocol achieve privacy-preserving requirement of some nodes in the network, as opposed to all nodes, that are included in the aggregation set by a hop count parameter from the initiating node. This way, both of the protocols require no information of overall network structure and privacy-preserving data aggregation is achieved by being able to communicate with direct neighbors of each node in the network only.

Preface

Finishing this thesis is the most challenging task that I have ever done in my life. This is written in most downhill phase of me, both physically and mentally. I'm glad that environment in the Netherland, Delft University of Technology especially, help me a lot to overcome my problem.

First of all, I would like to thank Zeki Erkin, my thesis supervisor, for guiding me in months of thesis period. Thank you for the insight given and for keeping me in track of what to see and explore in the topic of the thesis. And most importantly, thank you for understanding the hurdle that I had in early part of working on the thesis and for believing in me that I could overcome that. My thanks also go to PhD candidates and masters student of Cyber Security Delft University of Technology especially ones that also work under Zeki, thank you for not only providing professional academic environment but also fun leisure times. It awes me that you guys like to eat cakes that much. I also would like to thank Evangelos Pournaras of ETH Zurich for discussions and providing access to code repository that I use in this thesis.

Last but not least, I would like to thank Indonesia Endowment Fund for Education (Lembaga Pengelola Dana Pendidikan / LPDP) scholarship for giving me a chance to pursue my dream in experiencing higher education in Delft University of Technology.

Prahesa Kusuma Setia
Delft, November 2017

Contents

1 Introduction

The best way to keep our data inaccessible is to keep them in a safe environment where no single entity can interact with them. However, this approach renders our data useless as data is regarded as valuable assets if we can do some operation and gain some insight on them. Trusting other parties to handle our private data is also not the best approach in many cases as they can misuse our data for their own best interest. Developing a way to keep both our data as private as possible while still allowing other entities to operate on them and gaining value for all interested parties is currently a compelling topic.

In European Union (EU) privacy of one individual remains one of the critical aspects of society. Under EU regulation, personal data can only be gathered legally under a set of standard and rules [17]. Latest reform of data protection rules [40] also already been taken into force on May 2016 and will be widely applied throughout EU in May 2018 [9]. Not only happen in the European Union, but this consideration of privacy also being addressed in Canada [7] and the United States [25]. This concern about privacy should be taken into account when designing a system that handles sensitive data while at the same time requires a function to be computed over those mentioned data.

The simplest, albeit non-private, way to compute a function over all secret values of all parties participating in the system is to choose a trusted entity. This trusted entity collects all values from all of contributing parties and computes a function over them, and then publish the result to every party. If we could guarantee the privacy and authenticity of data transmission and we could guarantee that our trusted entity will behave as it is and not corruptible, then we already achieve our desired goal. A problem arises when we want to change our trust model not to trust anyone, not even trusting aforementioned trusted entity. Can we still achieve our goal in computing a function over all of the participant's secret values without requiring them to share those values? This kind of problem definition is what brings our interest in studying privacy-preserving data aggregation.

Privacy-preserving data aggregation is one instance of Multiparty Computation (MPC) problem (early problem of MPC studied in [48], other example applications include secure comparison in online auction [12], privacy-preserving face recognition [20], private data mining, private voting, private matchmaking [11]). We could define parties involved in privacy-preserving data aggregation as two separate types: node and aggregator. Typically, each node holds some private value that they want to keep as a secret while in the same time, they want to compute some function over all of other node's value. In this case, an aggregator may or may not help the nodes to compute the function. At the end of the protocol, however, the aggregator and other nodes should not have knowledge of any individual node's secret value.

Research in privacy-preserving data aggregation is conducted widely in numerous fields. Each of them uses it with slight modification according to individual needs in the field. However, their goals are similar. They are all in need to compute specific value while keeping individual value private. We briefly mention applications of privacy-preserving data aggregation in the field of power-grid and smart meter, wireless sensor network, and crowdsensing. Furthermore, we also discuss the similarity among them and what are the current protocols lacking.

1.1. Power-grid and Smart Meter

The current existing approach in calculating energy consumption in a neighborhood in a power grid consists of calculation in individual household involved in them. This mechanism plays a vital role in the power grid to dynamically provide and load balance energy. However, measuring individual electricity consumption in a household is privacy intrusive. Type of electrical appliances in a house can be deduced even by eye inspection of the power usage chart [27]. In Figure 1.1 we show an example of electricity usage of one household, from the figure we can indeed roughly deduce that particular house's hourly and weekly activity just by looking at the plot of the electricity usage. More advanced data mining and profiling techniques also enable long-term and detailed insight of house's activity [26] [39]. This knowledge could reveal a pattern that can be abused so that we can conclude what type of electrical appliances are available in the household or conclude whether a household has a friend staying overnight [24]. Concern about privacy should be taken into account in designing smart power-grid along with its smart metering mechanism. Growing energy needs allows governments to deploy a way to maximize existing power-grid infrastructure to include new energy resources generation and also monitor energy consumption on an individual grid.

Ideally, according to the EU direction, private data may only be collected and processed under stringent conditions and only if it is necessary to provide services [9]. Most of the time, parties that collect data about electricity usage in a power-grid only need the aggregated usage of several houses only, not individual usage, for them to provide services. Dynamically allocate power to the network and load-balancing the supply-demand is the example of services required. Several works have been carried upon in order to address this problem by utilizing cryptography techniques to achieve privacy while still allow required services. We study different proposed protocols that utilize additive homomorphism and secret sharing [24], additive homomorphism and Diffie-Helman key exchange [34] [19], differential privacy [2], signature and commitment scheme [41], Shamir threshold secret sharing [30], and Paillier homomorphic encryption [18].

1.2. Wireless Sensor Network

Application of privacy-preserving data aggregation also found its use in the field of wireless sensor networks (WSN). WSN is the type of network that consists of small devices with limited computing capabilities and energy storage. These small devices in the WSN typically share a common goal to compute a particular value based on the

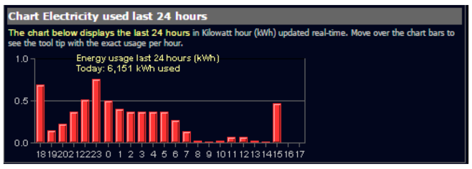
Chart Electricity used last 24 hours

The chart below displays the last 24 hours in Kilowatt hour (kWh) updated real-time. Move over the chart bars to see the tool tip with the exact usage per hour.

(a)

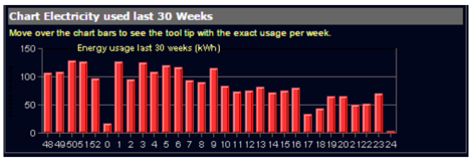
Chart Electricity used last 30 Weeks

Move over the chart bars to see the tool tip with the exact usage per week.

(b)

Figure 1.1: Electricity usage of one household in (a) 24 hours and (b) 30 weeks. Using this knowledge one can deduce the house's activity. Taken from [1].

reading of the environment that they are installed in, for example computing temperature, rainfall intensity, and geological activity.

Several works have been conducted for the security of WSN. Earlier works focused on key management [21], authentication [49], and message routing [31]. More recent research also address the security of data transmission in WSN by utilizing identity-based signature technique [36]. A survey paper also addresses a different type of attack that might occur in WSN at different layers of the network and what countermeasure needs to be done in order to overcome the different type of attacks [13].

In this thesis, we focus more on studying WSN capabilities in aggregating values over all of the nodes within the network. Due to the inherent limitations that exist in WSN, doing computation with less as possible and doing data transmission with less bit as possible are required features of data aggregation in WSN context. In the general scenario, the WSN usually deployed in a publicly accessible and un-trusted environment so that privacy also adds to the list of required feature to have, adding more issues to the hardware limitation of WSN. An efficient of privacy-preserving data aggregation using only simple mechanism by means of addition modulo is then explored in [6]. An extension of this mechanism is also explored in [2] by adding *differential privacy*, that is making each entry indistinguishable from other entries in the same networks. However, it comes with the price of the accuracy of the overall aggregation value.

1.3. Crowdsensing

Crowdsensing is a paradigm that arises from currently developed technology in our mobile devices. These devices not only constantly does communication with external parties but also can utilize their readily available sensors such as GPS, gyroscope, accelerometer that are embedded on them. By reading values from these sensors, we have the ability to acquire knowledge from individual's environment.

In a recent publication [23], the authors highlight the term *mobile crowdsensing*. They are the type of specific devices that rely on data collected from a large number of devices. The authors also divide crowdsensing into two main categories based on the involvement of the individual that own the devices, that is participatory sensing [4] or opportunistic sensing [35]. In the participatory sensing, the user takes an active role in providing the sensor data to the system, it can be in the form of the picture or video being taken, the review of a traffic incident, or location check-in in the location-based social network. On the other hand, the opportunistic sensing relies on automatic process and require less user intervention, for example periodically updating the location of the user's device.

Similar to the context of smart-meter that we already discussed earlier, sensing data that are obtained from the device might contain sensitive information of a user. For example, it might contain user's habit, location history, and favorite places. Sending this kind of information to the outside system is privacy intrusive, even more, when the user has less involvement in agreeing to participate.

Research in the specific field of crowdsensing to achieve privacy has already been conducted. In [45] the authors work on data aggregation in a privacy-preserving manner when the data is incomplete. The authors use bilinear mapping techniques along with matrix completion algorithm to treat the incomplete data.

1.4. Similarity in Different Fields

Looking at numerous application of privacy-preserving data aggregation in specific domains, we are interested in looking for similarity and comparing each one of them. In abstract, we can regard each protocol as interacting parties that want to compute aggregation value (that is, the summation of each value), we refer them as *node*. This can further be achieved with or without the help of additional parties that specifically only do the calculation. We refer them as *aggregator*. However the overall goal of them are the same, that is to also achieve privacy so that each value should not be known by the others.

All mentioned works earlier differ in the underlying network structures. This is due to the existence of *aggregator* or not in them and also due to the how each *node* interacts with others. They also differ in the choice of cryptography primitive used, whether they use homomorphic encryption technique and secret sharing to allow private data aggregation, or use differential privacy to provide indistinguishability between an entry against each other. It is therefore interesting to contrast each of them more in this sense.

In a dynamic scenario of the network, where there might be a new node being added or disappear in case of node or communication failure, doing data aggregation is not a trivial work to do. Computing a certain value in the currently available or even some of the selected node within the network is the goal. Moreover, a new challenge arises if we consider achieving privacy in the network. Current research in handling the dynamic number of node in the network when doing privacy-preserving data aggregation is available in the context of smart meter that tolerates failures [18] [30]. However, they only work under synchronous network communication assumption. Having a protocol that can handle the dynamic scenario of the network while still maintaining privacy is the goal of our research.

1.5. Peer-to-Peer Network

In this thesis, we explore the usage of peer-to-peer network paradigm to implement privacy-preserving data aggregation protocols. A peer-to-peer network is a network structure paradigm where every node in the network communicates with each other without a central authority. It differs with client-server or centralized network structure where the existence of a server or central authority is needed in the network. Due to this property of having no central authority that is trusted by nodes, security and privacy are inherently important in a peer-to-peer network. Several research in the field of peer-to-peer network usually focused on file-sharing [8] [42] [3], secret-sharing [47], anonymous routing for anonymity [16], anonymous storage system [15], etc.

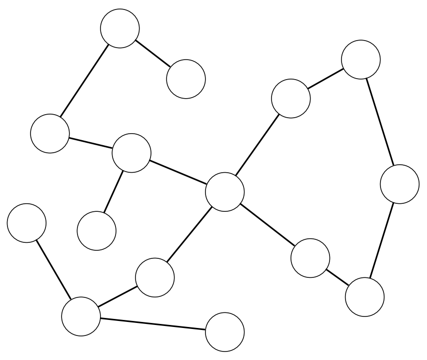
Figure 1.2: Prior works assume a total knowledge and non-dynamic structure of the network

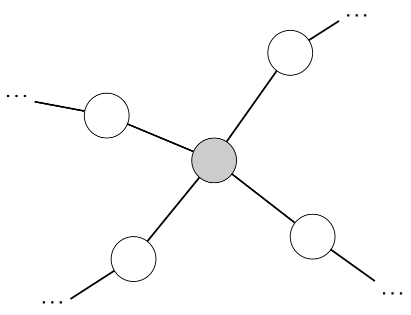

Figure 1.3: Generally a node in the network only have knowledge of its direct connection and not the whole network

The inherent characteristics of the paradigm that treats every node in the network as an independent entity makes trust delegation in the peer-to-peer network, not a viable option, as every node in the network should not trust any other node. Moreover, in peer-to-peer network paradigm, every node in the network does not have knowledge of the whole network structure, they only know their direct neighbor connection. If every node in the peer-to-peer network has a secret value and they want to compute certain aggregate function together without having to trust any other node, then having a data aggregation protocol in a privacy-preserving way is the desired goal. Modeling the protocol as an interaction of nodes that have various states in each protocol execution as opposed to treating the network as a whole is our focus.

Questions might arise of the need and applicability of privacy-preserving data aggregation in peer-to-peer network scenario. Indeed, in most cases, it is easier to have a centralized network structure where every node report to a central authority rather than having a peer-to-peer scenario where every node could act independently. In this section, we explore one of the real-world problem that motivates us to do this research, and why it is important to have privacy-preserving data aggregation in peer-to-peer network scenario.

The latest advancement in the technology of power system allows a shift of paradigm of the role of an end user to not only be a consumer of power but also become a power producer. A house that has a 3kW solar panel installed produces enough energy to power electricity consumption of the household [46]. Having more solar panel installed, or having other power generator installed means the house produces more electricity than they need, thus they could become power producer themselves. This further allows a decentralized power grid network structure to be built.

Characteristic of decentralized power grid network is that the power is generated near where the demand is [32]. This is by far different with a centralized power system that power is generated in a central system and then distributed to the number of consumers that are far away. This feature that states the power is generated near where the demand is could cut the cost of electricity distribution. As a centralized power system requires heavy investment in the infrastructure and the power be produced by non-renewable energy sources [29].

Authors in [5] propose a framework of the distributed power system where there is no single power producer. Instead, several houses in a neighborhood could both become a power producer and consumer, named a prosumer. The paper also describes a simulation model of how to organize automated trading and pricing of electricity usages and also load balancing. The proposed framework works by separating roles and steps into four different layers based on the process being conducted.

However, a current state of research in the field of decentralized power grid lack focus on the privacy concern of the end user. By having each of the houses in the neighborhood could both act as a power producer and power consumer, a mechanism to both track the power being consumed in the neighborhood and ensure the privacy of each house in the neighborhood is a required feat as stipulated by the regulation.

1.6. Research Goal

We already discussed our motivation in researching privacy-preserving data aggregation in peer-to-peer network scenario. We also showed that our research topic has at least one real-world scenario where it could be applied on. Our focus in this thesis, however, is to not focus only on one specific domain of implementation. Rather, we aim to formulate our problem definition and propose our solution in generic concept. The underlying research question of this thesis is then as follow:

> *How can privacy be achieved in data aggregation of peer-to-peer network scenario where there is no authority and each node in the network has no knowledge of the overall network structure?*

From this research question, several sub-questions arise:

1. How can data of the secret values of each of the node be aggregated in the network without leaking the values themselves?

2. How does dynamic structure of the network where nodes can be added or deleted be managed?

3. How can we achieve data aggregation in a privacy-preserving way with each node knows nothing about overall network structure?

4. How can data aggregation be achieved in a privacy-preserving way without single central authority?

5. What cryptographic primitive works best in term of privacy, operation time, and message transmission's size of the network?

1.7. Research Contribution

To address the research question and sub-questions, in this thesis our contribution is to present two protocols of privacy-preserving data aggregation in a peer-to-peer network. The first protocol utilizes additive homomorphism properties of Paillier cryptosystem and the second protocol utilize additive secret sharing. The second protocol runs approximately 23 percent faster in the case of 5,000 nodes compared with the first one in term of runtime required in order to complete one phase of aggregation step, although it does require more assumption in the network structure (explained more in Chapter 5).

Both of our proposed protocols require no knowledge of overall network structure. Instead, a node in the network communicates only with its direct neighbors. They both work under honest but curious security model, that is every party in the network assumed to execute the protocol steps honestly but they might learn more based on the data that they obtained. Both of them also fulfill requirements of being *correct* and *privacy preserving*.

1.8. Document Outline

The structure of this document is as follow. The first chapter discusses background and use cases of where privacy-preserving data aggregation in peer-to-peer network scenario is important. Chapter 2 explores prior existing work in the field of privacy-preserving data aggregation. We also study eight different protocols by focusing on their shared similarity, not on their specific application domain. In Chapter 3 we discuss our research methodology and the goal of our proposed protocols formally. Chapter 4 and 5 both discuss our proposed protocols in details, the first one utilizes Paillier and the second one utilizes secret sharing. We also provide analysis of each protocol in term of correctness, privacy-preserving, and complexity in their respective chapter. Chapter 6 discuss implementation and experimental result of both protocols. Finally, in Chapter 7 we provide discussion and future work opportunities.

2 Prior Works

The topic of privacy-preserving data aggregation has been widely studied in numerous contexts. In this chapter, we review the latest studies in this field while focusing on key features that are common to all of them. When giving the brief review in this chapter, we focus on the underlying network model, cryptography primitives used, and some remarks regarding advantages and disadvantages of each protocol compared with the others.

We consider each protocol as a particular instance of secure multiparty computation where several players want to compute specific value together privately. There are two types of players: *node* and *aggregator*, we use this terminology when addressing players in protocols being reviewed regardless of their specific roles in them. *Node* is a type of players that provide value in the protocol and may or may not involve in computation process. *Aggregator* is a type of players that obtain values from *node*s and conduct computation on them. The goal of privacy-preserving data aggregation is to compute certain value over all values provided by *node*s while maintaining the privacy of each *node* value, that is individual node value should not be known either by *aggregator*s or other *node*s.

2.1. Preliminaries

Before giving more details of each protocol reviewed, we give a brief overview of basic preliminaries required to understand the protocols.

2.1.1. Homomorphic Encryption

We formalize our definition of an encryption scheme that is homomorphic (thus we can refer it as *homomorphic encryption*) as follow.

Let $\oplus$ and $\otimes$ be binary operators operating on two operands. Let $\mathcal{E}()$ denote an encryption scheme. Let $m_1, m_2 \in \mathbf{M}$ denote plaintexts that are sampled from message space $\mathbf{M}$ satisfying a group under operation $\oplus$. Let $c_1, c_2 \in \mathbf{C}$ denote ciphertexts that are sampled from ciphertext space $\mathbf{C}$ satisfying a group under operation $\otimes$. If we have $c_1 = \mathcal{E}_k(m_1)$ and $c_2 = \mathcal{E}_k(m_2)$ for existing encryption key k, then $\mathcal{E}()$ is a *homomorphic encryption* if it satisfies,

$$\mathcal{E}_k(m_1 \oplus m_2) = c_1 \otimes c_2. \tag{2.1}$$

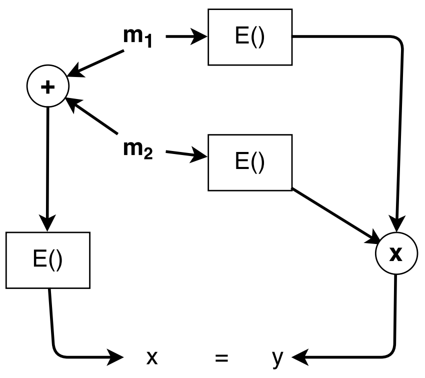

Figure 2.1: An illustration of how homomorphic encryption work. Let the circle in the figure represent two different binary operator $\oplus$ and $\otimes$ respectively, then the resulting x is equal to the resulting y.

As can be seen in Figure 2.1, any player can compute $c_1 \otimes c_2$ without having to know the encryption key k of them. The resulting value will be the encryption of $m_1 \oplus m_2$. This characteristic of encryption is suitable if we want to delegate some computation to an untrusted player so that they will not be able to see individual messages but they can compute on them. We sometimes refer a *homomorphic encryption* scheme as *additively homomorphic* if $\oplus$ is an addition operator (for example, if we let our **M** equal to $\mathbb{Z}$ or $\mathbb{Z}_p$).

2.1.2. Pailier Encryption Scheme

Pailier encryption scheme [37] is one instance of *additively homomorphic* encryption. The scheme works as follows.

Let $m \in \mathbb{Z}_N$ be a plaintext, where N is a product of two large prime numbers. Let pk be a public key in the form of a tuple containing (g, N) where g is a generator of order N. Let r be a randomly sampled element of $\mathbb{Z}_N^*$. Pailier encryption defined as:

$$\mathcal{E}_{pk}(m) = g^m \,.\, r^N \bmod N^2. \tag{2.2}$$

Suppose that we have $c_1 = \mathcal{E}_{pk}(m_1)$ and $c_2 = \mathcal{E}_{pk}(m_2)$. Then we have

$$c_1.c_2 = g^{m_1}.r_1^N.g^{m_2}.r_2^N \bmod N^2, \tag{2.3}$$
$$c_1.c_2 = g^{(m_1+m_2)}.(r_1 r_2)^N \bmod N^2, \tag{2.4}$$
$$c_1.c_2 = \mathcal{E}_{pk}(m_1 + m_2), \tag{2.5}$$

that shows us a multiplication of two ciphertexts will result in an encryption of an addition of two plaintexts, thus indeed Pailier scheme is *additively homomorphic*. The decryption process works as explained well in [37].

2.1.3. Chinese Remainder Theorem

In [18], chinese remainder theorem described as follows. Let $p_1, p_2, ..., p_r$ denote relatively prime positive integers, that is $gcd(p_i, p_j) = 1; i, j \in \{1, 2, ..., r\}; i \neq j$. Let $a_1, a_2, ...,$ a_r be integers. The system of r congruences

$$x \equiv a_i \bmod p_i\,;\; i \in \{1, 2, ..., r\}, \tag{2.6}$$

has a unique solution modulo $P = p_1 p_2 ... p_r$ as

$$x = \sum_{i=1}^{r} a_i P_i y_i \bmod P, \tag{2.7}$$

where $P_i = P/p_i$ and $y_i = P_i^{-1} \bmod p_i; i \in \{1,2,...,r\}$.

2.1.4. Differential Privacy

In [2], A function $\mathcal{F}$ is ϵ-differential private, if for all data sets D_1 and D_2, where D_1 and D_2 differ only in single entry, and for all subset of possible answers $S \subseteq Range(\mathcal{F})$. Formally,

$$P(\mathcal{F}(D_1) \in S) \le e^{\epsilon} P(\mathcal{F}(D_2) \in S). \tag{2.8}$$

It says that differentially private function produce indistinguishable outputs for inputs that differ by a single entry. Modification of any single entry in the dataset, including deletion or addition, changes the probability of any output up to a multiplicative factor of e^{ϵ}.

To achieve *differential privacy*, we can use the following mechanism. For all $f : \mathbb{D} \to \mathbb{R}^r$, we can re-construct f by adding noise that is carefully calibrated to the global sensitivity of f such that $f(D) \leftarrow f(D) + \mathcal{L}(\epsilon)$, where $\mathcal{L}(\epsilon)$ is independently generated random variable following Laplace distribution with ϵ parameter.

We can also notice that Lapplacian noise mentioned above is infinitely divisible, that is

$$\mathcal{L}(\epsilon) = \sum_i (\mathcal{G}_1(i,\epsilon) - \mathcal{G}_2(i,\epsilon)), \tag{2.9}$$

where $\mathcal{G}_1(i,\epsilon)$ and $\mathcal{G}_2(i,\epsilon)$ denotes i.i.d random variables having gamma distribution with a certain probability density function.

2.1.5. Threshold Secret Sharing

Threshold secret sharing, or sometimes called *Shamir's secret sharing*, lets us recover a secret from n number of shares as long as d number of them contribute to it [11][30][43]. It utilizes a property of *Lagrange interpolation*.

For a set $C \subseteq \{1,2,...,n\}$ and a finite field $\mathbb{Z}_p$ with p is a prime and $p > n$, *Lagrange polynomials* $\lambda_i^C(x)$ is defined as

$$\lambda_i^C(x) = \prod_{t \in C \setminus \{i\}} \frac{x-t}{i-t}, \tag{2.10}$$

and *Lagrange coefficient* as $\lambda_i^C(0)$. Then, for any polynomial $P(x)$ over $\mathbb{Z}_p$ with degree at most $|C| - 1$ we have

$$P(x) = \sum_{i \in C} P(i) \lambda_i^C(x), \tag{2.11}$$

and especially

$$P(0) = \sum_{i \in C} P(i) \lambda_i^C(0). \tag{2.12}$$

Protocol	Network Structure	Cryptography Primitive
Castellucia, Mikletun, Tsudik [6]	Tree structured network, aggregator in the root	Noise addition of k (mod M), ciphertext is additively homomorphic
Garcia and Jacobs [24]	N nodes with one aggregator	Additive homomorphic encryption and secret sharing
Kursawe, Danezis, Kohlweiss [34]	N nodes with one aggregator	Additive secret sharing, Diffie-Helman key exchange, bilinear map, bruteforcing
Ács and Castelluccia [2]	N nodes with one aggregator	Differential privacy
Corrigan-Gibbs and Boneh [10]	N nodes with M aggregators	Arithmetic circuit non-interactive proof
Erkin [18]	Even number of nodes with one aggregator	Chinese remainder theorem and Pailier
Erkin and Tsudik [19]	Peer-based nodes	Modified Pailier homomorphic encryption
Hoepman [30]	Peer-based nodes	Shamir threshold secret sharing

Table 2.1: Privacy-preserving data aggregation protocols comparison

Threshold d-out-of-n secret sharing is based on this properties. Let $s \in \mathbb{Z}_p$ be a secret to be shared, we sample $\beta_1, ..\beta_{d-1}$ randomly from $\mathbb{Z}_p$ and define $P(x)$ to be randomly chosen $(d-1)$ degree secret sharing polynomial over $\mathbb{Z}_p$

$$P(x) = s + \sum_{i=1}^{d-1} \beta_i x^i, \tag{2.13}$$

such that $P(0) = s$. We then can generate n secret shares as $s_i = P(i)$. Any k number of such n shares can be used to reconstruct s, for $k \geq d$. Let $C = \{c_1, .., c_k\}$ denotes a chosen such index of secret shares. Then we have

$$\sum_{i \in C} \lambda_i^C(0) s_i = P(0) = s. \tag{2.14}$$

2.2. Privacy-Preserving Data Aggregation Protocols

In this subsection, we mention related privacy-preserving data aggregation protocols in details. A brief comparison can be found in Table 2.1.

2.2.1. Tree Structured Nodes Using Additive Homomorphism Noise

Castelluccia, Mykletun, and Tsudik proposed a protocol where the *node*s are structured in a tree with an *aggregator* resides in the root [6]. The protocol uses a simple addition of noise to encrypt each *node* value thus resulting in a simple and lightweight operation.

Messages are represented as integer m modulo M where M is large integer, that is $m \in [0, M-1]$. To encrypt a message, a node randomly choose $k \in [0, M-1]$ and compute c, the encryption of message as

$$c = \mathcal{E}_k(m, M) = m + k (\mathrm{mod}\, M). \tag{2.15}$$

This addition of k can be observed as adding a *noise* acting as a key to the message. Decryption of ciphertext works simply as

$$\mathcal{D}_k(c, M) = c - k (\mathrm{mod}\, M). \tag{2.16}$$

The resulting ciphertexts are *additively homomorphic*, we can add two of them together and the result will be the encryption of messages addition. In other word, for messages m_1 and m_2 let $c_1 = \mathcal{E}_{k_1}(m_1, M)$ and $c_2 = \mathcal{E}_{k_2}(m_2, M)$, we then have

$$\mathcal{D}_{k_{sum}}(c_1 + c_2, M) = m_1 + m_2, \tag{2.17}$$

for $k_{sum} = k_1 + k_2$.

This characteristic of resulting ciphertext can be utilized in a tree structure. In proposed protocol, we have a tree of *node*s with one *aggregator* in its root. The leaves of the tree are a collection of *node*s that provide values. Each *node* forward an encryption of its value to its parent, due to the nature of *additively homomorphism*, its parent can then do addition over all of its received ciphertexts, resulting in a new ciphertext of encryption of all its child's value. Since each *node* only have knowledge of its own encryption key, other *node* cannot obtain the original value.

The root of the tree, which is the *aggregator*, will then obtains all ciphertext of its children. It can then add them all together to obtain encryption of the total value of all `nodes`. The only downfall of this protocol is that since it requires the sum of all keys to do decryption, the *aggregator* need to have an ability to know each of leaves' encryption key. This can be of course can be generated easily using a pseudo-random function with each node's id acting as a seed.

2.2.2. N Nodes With One Aggregator Using Homomorphic Encryption and Secret Sharing

Garcia and Jacobs proposed a protocol where there are n number of *node*s with one *aggregator*. The protocol uses IND-CPA secure *additive homomorphic* encryption and simple secret sharing [24]. Combination of those techniques is required so that the *aggregator* and other *node*s will only see random shared values or the sum of all values on any given time in protocol runtime. The authors pick Pailier [37] as the protocol's encryption scheme, but we can pick any scheme as long as it satisfies $\mathcal{E}_k(m_1) . \mathcal{E}_k(m_2) = \mathcal{E}_k(m_1 + m_2)$.

The protocol works as n nodes, denoted by $P_1, P_2, ..., P_n$, each having a value $m_1, m_2, ..., m_n$ and associated with publicly known public key $pk_1, pk_2, ..., pk_n$ respectively, want to compute $\sum_{j=1}^{n} m_j$. Each of P_i picks n random numbers denoted by $a_{i,j}$ so that each holds

$$P_i : m_i = a_{i,1} + a_{i,2} + ... + a_{i,n} \text{ (mod } \lambda), \tag{2.18}$$

for a large integer λ and then construct

$$y_{i,j} := \mathcal{E}_{pk_j}(a_{i,j}) \; ; \; j \in \{1, .., n\} \setminus \{i\}, \tag{2.19}$$

P_i sends $y_{i,j}$ to the *aggregator*. Note that $y_{i,i}$ does not exist by construction, P_i keeps $a_{i,i}$ for itself.

After receiving the encrypted values, the *aggregator* can compute

$$\prod_{j\neq i} y_{j,i} = \mathcal{E}_{pk_i}(\sum_{j\neq i} a_{j,i}) \pmod{\lambda}. \tag{2.20}$$

for all $i \in 1,...,n$. The *aggregator* then sends $\mathcal{E}_{pk_i}(\sum_{j\neq i} a_{j,i})$ to P_i.

Each P_i of course can decrypt the encryption to obtain $\sum_{j\neq i} a_{j,i}$. They then can add $a_{i,i}$ that they hold themselves so that

$$\sum_{j\neq i} a_{j,i} + a_{i,i} = \sum_{j} a_{j,i} \pmod{\lambda}, \tag{2.21}$$

this value is then sent again to the *aggregator*. After receiving values from P_i *aggregator* can finally compute total sum as

$$\sum_i \sum_j a_{j,i} = \sum_{j=1}^{n} m_j \pmod{\lambda}. \tag{2.22}$$

This protocol uses a simple yet well-known Pailier scheme to achieve its goal. However, this protocol requires that all *node*s behave normally in every step of protocol run. One *node* failure will fail the protocol altogether because there is no way to obtain the final result without all *node*s participating normally. It also requires that all *node*s and *aggregator* operates in an authenticated channel, meaning there is a guarantee that all values sent are indeed come from the said player.

2.2.3. N Nodes With One Aggregator Using Masking and Brute-Forcing

Kursawe, Danezis, and Kohlweiss proposed an n number of *node*s with one *aggregator* protocol utilizing masking techniques and brute-forcing [34]. They also considered several realizations of the protocol to derive shared randomness, namely using secret sharing, Diffie-Helmann key exchange, and a bilinear map. The protocol comes with an assumption that *aggregator* knows roughly the total value, in other words, *aggregator* needs to know the distribution of the total value for this protocol to work. They also consider two types of protocol, *aggregation protocol* and *comparison protocol*.

In *aggregation protocol*, each node P_i that has value m_i uses masking value k_i to output noised value $m_i + k_i$. The only requirement for masking values are they have to be generated so that the sum of them all set to zero, that is $\sum_i k_i = 0$. The result of the protocol then be

$$\sum_i (m_i + k_i) = \sum_i m_i, \tag{2.23}$$

by summing all nodes' noised values.

In *comparison protocol*, each node P_i output $g^{m_i+k_i}$ instead. The result of the protocol then be

$$\prod_i g^{m_i+k_i} = g^{\sum_i (m_i+k_i)} = g^{\sum_i m_i}, \tag{2.24}$$

by multiplying all nodes' output, where the value of g is known by all parties involved. This protocol called comparison protocol because the *aggregator* needs to know roughly the total sum and use this protocol to determine whether the sum of protocol is close enough to its rough knowledge of it. Suppose that *aggregator* knowledge of rough total sum is m_{guess}, it can compute $g^{m_{guess}}, g^{m_{guess}+1}, g^{m_{guess}-1}, ...$ until a match is found or until the difference becomes too big and decide that it is an anomaly.

The core of these two protocols lies in the generation and distribution of k_i so that their sum is zero and g^{k_i} so that their product is one. The authors propose four methods in order to generate those values, one using secret sharing, three using Diffie-Hellman key exchange and a bilinear map. In here we briefly give the summary of the first two methods, for the two other methods are already explained well in [34].

The first method using secret sharing works by selecting subset of n nodes as *leaders* with size p, denoted by index $\ell_1, \ell_2, ..., \ell_p$. Note that $\{P_{\ell_1}, P_{\ell_2}, ..., P_{\ell_p}\} \subset \{P_1, P_2, ...P_n\}$. Firstly, each node P_j computes p random values $s_{j,1}, s_{j,2}, ..., s_{j,p}$. They then send each corresponding $s_{j,k}$ to P_{ℓ_k} (this also can be achieved by relaying it through *aggregator* by firstly encrypt it with each leader's public key). Each leader P_{ℓ_k} collects $n-1$ shares $s_{j,k}, j \in \{1,..,n\} \setminus \{\ell_k\}$ and compute its own share $s_{\ell_k,k}$ such that all shares summed to zero. Finally, all *node*s add all their shares $s_{j,1}, s_{j,2}, ..., s_{j,p}$ to get their main share s_j. This value can be used for k_i both in aggregation and comparison protocol.

Second method uses Diffie-Helman key exchange protocol. Let sk_i and pk_i denote secret key and public key of *node* P_i. Choose g so that it be a generator of Diffie-Hellman group, note that g value is the same in all P_is. Each P_i compute public key as $pk_i = g^{sk_i}$ and distribute it to all other *node*s by first certifying it. After verifying all received value, all *node*s can compute

$$g^{k_i} = \prod_{j \neq i} pk_j^{\tau(j,i)\, sk_i}, \tag{2.25}$$

$$= \prod_{j \neq i} (g^{sk_j})^{\tau(j,i)\, sk_i}, \tag{2.26}$$

where

$$\tau(j,i) = \begin{cases} -1, & j < i, \\ 1 & else. \end{cases} \tag{2.27}$$

We then can show that the sum of all k_is is zero

$$\sum_i k_i = \sum_i \sum_{j \neq i} \tau(j,i)\,.\, sk_j\,.\, sk_i = 0. \tag{2.28}$$

2.2.4. N Nodes With One Aggregator Using Differential Privacy Noise

Ács and Castelluccia propose an n nodes protocol and one *aggregator* using differential privacy noise [2]. We can consider this protocol as an improvement of [6] (please refer to 2.2.1), because this protocol use a same *additively homomorphic* encryption scheme as in the latter protocol.

This protocol adds *differential privacy* noise in the secret value of each *node* before doing encryption step on them (please refer to 2.1.4). It can be noticed that since Lapplacian noise can be divided many times infinitely, they can be generated distributively by all participating *node*s. Therefore all *node*'s secret value m_i is added by noise so that it becomes,

$$m_i \leftarrow m_i + \mathcal{G}_1(i,\epsilon) - \mathcal{G}_2(i,\epsilon). \tag{2.29}$$

When we aggregate all of them, the result will yield to

$$\sum_i m_i + \mathcal{L}(\epsilon), \tag{2.30}$$

which is ϵ-differential private.

After adding Lapplacian noise to each m_i value, node P_i encrypt it with *additively homomorphism* scheme as described in 2.2.1 with some extension described below, resulting in ciphertext c_i. Note that the secret key needs to be known by *aggregator*, so that let $K_{i,Agg}$ denotes secret key that is known by P_i and *aggregator*. Then, P_i selects ℓ other *node*s randomly so that if P_i selects P_j then P_j also selects P_i. They then agree on a dummy key $r_{i,j} = (-1) . r_{j,i}$ shared between them. The encryption process then becomes,

$$c_i = m_i + \sum_j r_{i,j} + K_{i,Agg} \bmod M, \tag{2.31}$$

and each of them being sent to the *aggregator*. *aggregator* will not know individual value because we construct the encryption to have random noise that can only be nullified if it total all of them together.

After receiving all c_i, the *aggregator* can just total them together, nullifying the $r_{i,j}$ part because $\sum_i \sum_j r_{i,j} = 0$. Because it knows all of $K_{i,Agg}$ values, it can just subtract them to retrieve final total value. Formally,

$$\begin{aligned} \sum_i (c_i - K_{i,Agg}) &= \sum_i (m_i + \sum_j r_{i,j} + K_{i,Agg} - K_{i,Agg}), && (2.32)\\ &= \sum_i m_i + \sum_i \sum_j r_{i,j}, && (2.33)\\ &= \sum_i m_i. && (2.34) \end{aligned}$$

Note that we need all *node*s to be participated according to protocol in order for *aggregator* to be able to recover final sum, therefore this protocol cannot handle *node* failure.

2.2.5. N Nodes With M Aggregators Using Arithmetic Circuit Non-Interactive Proofs

Corrigan-Gibbs and Boneh propose a protocol that consist of n number of *node*s with m number of *aggregator*s using arithmetic circuit non-interactive proofs [10]. The rationale of needing m *aggregator*s is to provide privacy as long as one of them is honest and working correctly, while the correctness of the protocol can only be achieved when all of them all honest and working correctly.

The authors provide a simplified overview of the protocol that works as follows. Suppose we have n *node*s denoted by P_i, each of them holds a value x_i and want to compute $\sum_i x_i$. We also have m number of *aggregator*s denoted by Agg_j. Let λ be a prime number acting as a public parameter. The protocol then works in three steps:

1. **Create Shares** Node P_i splits its value x_i into m number of shares according to constraint $x_i = a_{i,1} + a_{i,2} + ... + a_{i,m} \pmod{\lambda}$. Each P_i then sends one share for each Agg_j over an authenticated channel.

2. **Aggregate** Each Agg_j holds an accumulator scalar A_j initialized to zero. Upon receiving $a_{i,j}$ from previous step, each of them update the accumulator as $A_j \leftarrow A_j + a_{i,j}$.

3. **Publish Result** After all Agg_j receive all shares, they publish their A_j values so that other *aggregator*s can retrieve it. Computing $\sum_j A_j$ will yield desired result $\sum_i x_i$.

The simplified overview of the protocol provides privacy because all *aggregator*s learn the sum of x_i while learning nothing about the initial value of the *node*s. We also can notice that it does not provide *robustness*. A single corrupt *node* can render the protocol corrupt by submitting any random integer to the *aggregator*s that not corresponds to its valid value.

The authors propose a method to achieve *robustness* in the protocol by using arithmetic circuit non-interactive proofs. Notice that in the **Aggregate** step each *aggregator* has no way of knowing whether the shares that they received are validly generated shares. In the proposed method, there exists a predicate $Valid()$ constructed using *arithmetic circuit* that used to check whether given share is a valid share or not without knowing the exact value of it. This predicate is given as public knowledge and known by all players involved. The *node*'s goal is to convince the *aggregator* that $Valid(x) = 1$ without giving any information about x. Each *node* needs to send a proof string to each *aggregator*, and the *aggregator*s gossips among other to conclude to accept x or not. The protocol achieves *zero knowledge* property and is called *secret-shared non-interactive proofs* or simply SNIP. The authors also describe on how to do data encoding in the protocol and how it can be used in combination with SNIP to achieve wide arrays of privacy-preserving data aggregation other than the sum, they are explained well in [10].

2.2.6. Even Number of Nodes With One Aggregator Using CRT and Homomorphic Encryption

Erkin proposed a protocol with even number of *node*s with one *aggregator* using Chinese Remainder Theorem (CRT) and *homomorphic encryption*, namely Pailer scheme [18]. The reason that the protocol needs even number of *node*s is that they are paired with another unique *node* and form groups. Therefore the protocol can tolerate *node* failures without disturbing all protocol runs, although one *node* failure will fail its pair. One of the advantages of this protocol is that with a single execution of it, *aggregator* can retrieve total consumption of the network as well as smaller groups that are created within it according to predefined criteria.

Suppose that we have K groups denoted with G_k for $k \in \{1,2,...,K\}$. Each of group has an even number of *node*s (with N is an even number denoting total number of *node*s in the network). In every group, each *node* denoted by H_i is paired with another unique *node* inside the group. We denote each pair as $G_{k,\ell}$, a pair inside G_k. The protocol works in the following steps:

1. **Setup** The network agrees on K prime numbers, each associated with group G_k and denoted by p_k. In each group, *node*s pairs are created. Each pair of *node* agree on a secret key $\alpha_{k,\ell}$, this can be achieved by using for example Diffie-Hellman key exchange protocol [14]. The *aggregator* generates key pairs of Pailier

scheme and publishes the public key. Because we give all primes p_k as public values, we also have $P = \prod_k p_k$. The network also agrees on $h \in \mathbb{Z}_n^*$.

2. **Reporting** Let tuple (g, n) denote Pailier public key associated with *aggregator*. Each H_i in $G_{k,\ell}$, having secret value m_i, perform these steps:

   (a) H_i computes
   $$m'_i = m_i P_i y_i \bmod P, \tag{2.35}$$
   just like defined in Chinese Remainder Theorem. Recall that $P_i = P/P_k$ and $y_i = P_i^{-1} \bmod p_k$.

   (b) H_i generates random number $r_i \in \mathbb{Z}_n^*$, and then calculate modified encryption function as
   $$\mathcal{E}_{pk}(m'_i) = c_i = g^{m'_i} . r_i^n . h^{n-\alpha_{k,\ell}} \bmod n^2, \tag{2.36}$$
   and sends c_i to *aggregator*.

   (c) Let H_j denote H_i's pair in $G_{k,\ell}$ and it has secret value m_j. It computes
   $$\mathcal{E}_{pk}(m'_j) = c_j = g^{m'_j} . r_j^n . h^{n+\alpha_{k,\ell}} \bmod n^2, \tag{2.37}$$
   similar with its pair, the only difference is in the power of h. It then sends c_j to *aggregator*.

3. **Aggregation** *aggregator* calculate aggregate of all input from *node*s by utilizing *additive homomorphism* properties.
   $$\mathcal{E}_{pk}(T) = \prod_{i=1}^{N} c_i = \mathcal{E}_{pk}(\sum_{i=1}^{N} m'_i). \tag{2.38}$$

4. **Total Computation** *aggregator* can decrypt $\mathcal{E}_{pk}(T)$ to reveal T, however this value is not the total consumption. T needs to be processed in order to reveal the total consumption T_k of group G_k as follows.
   $$T_k = \sum_{i \in G_k} m_i = T \bmod p_k. \tag{2.39}$$
   The total consumption of all groups is simply $\sum_k T_k$.

2.2.7. Peer-based Nodes Using Modified Homomorphic Encryption

Erkin and Tsudik propose a peer-based nodes protocol using modified *homomorphic encryption* [19] where it requires no involvement of separate *aggregator* in order to calculate the total sum of the values. In this protocol any *node* can act as *aggregator* and compute the aggregation result. The protocol uses a modified version of Pailier encryption scheme to achieve its goal, taking only the *additively homomomorphism* properties of the scheme. In fact, Pailier decryption key is made public knowledge in this protocol. The authors also consider three scenarios:

- **Spatial**: where we calculate the total value of all *node*s in one specific time instant.

- **Temporal**: where we calculate the total value of one *node* in several time period.
- **Spatio-Temporal**: the combination of two previous scenarios.

Recall that in Pailier, encryption is defined as

$$\mathcal{E}_{pk}(m) = g^m \,.\, r^n \bmod n^2. \tag{2.40}$$

The core idea of this protocol is to split modulo n of Pailer scheme into random N shares where N denote the number of *node* in the network. Let n_i denote the modulo shares received by node P_i. We need to have $\sum_i n_i = n$ in this protocol. This is because in each *node* the encryption will be

$$\mathcal{E}_{pk}(m_i) = g^{m_i} \,.\, r^{n_i} \bmod n^2, \tag{2.41}$$

where m_i denotes the *node*'s secret value. After encrypting each of their secret value, each *node* publishes the encryption so that each other can obtain every encryption. Then each of them can compute

$$\begin{aligned}\prod_i \mathcal{E}_{pk}(m_i) =& \prod_i (g^{m_i} \,.\, r^{n_i}) \bmod n^2, && (2.42)\\ =& g^{\sum_i m_i} \,.\, r^{\sum_i n_i} \bmod n^2, && (2.43)\\ =& g^{\sum_i m_i} \,.\, r^{n} \bmod n^2, && (2.44)\\ =& \mathcal{E}_{pk}(\sum_i m_i), && (2.45)\end{aligned}$$

thus all *node*s can decrypt it. Note that this protocol cannot handle node failure, one corrupt node will render the protocol useless because decryption result can only be obtained if all *node*s comply with the protocol. The authors consider using other trusted third party that in order to handle such failure to help retrieve corrupt *node*'s value, the only problem if we use this approach it means we delegate our trust to another party.

2.2.8. Peer-based Nodes Using Threshold Secret Sharing

Hoepman describe a peer-based nodes protocol using threshold secret sharing that can tolerate up to d predefined failures of n-sized *node*s [30]. The protocol works by utilizing a lemma that threshold secret sharing is additive. That is if we have two randomly chosen $(d-1)$ degree polynomials P and P' over $\mathbb{Z}_p$ each having secret $s = P(0)$ and $s' = P(0)$ respectively, then $s_i + s'_i$ is secret shares of $s + s'$.

The protocol works as follow. Let us decide a d number of minimum surviving *node*s that we expect to behave correctly in the protocol. It is the same way of saying we let maximum of t number of *node* failure, where $d = n - t$. Let m_i denote a secret value of *node* N_i. We then have these following steps in the protocol:

1. **Initialize** Each of surviving (non-faulty) N_i does
   - N_i construct randomly chosen $(d-1)$ degree polynomial P_i such that $P_i(0) = m_i$ using threshold secret sharing,
   - N_i sends secret shares $f_{i,j} = P_i(j)$; $j \in \{1,..,n\}$ to N_j.
2. **Aggregate Shares** Each of surviving N_j does

- N_j initialize $I_j = \varnothing$,
- N_j adds index i from received $f_{i,j}$, that is $I_j \leftarrow I_j \cup \{i\}$. Note that it only receive them if N_i from step 1 is a surviving *node*,
- N_j computes aggregate of secret shares that it received $F_j = \sum_{i \in I_j} f_{i,j}$,
- N_j publishes F_j to all *node*s.

3. **Calculate Sum** Each of surviving N_i does

   - N_i initialize $J_i = \varnothing$,
   - N_i adds index j from received F_j, that is $J_i, \leftarrow J_i \cup \{j\}$. Note that it only receive them if N_j from step 2 is a surviving *node*,
   - N_i compute aggregated value as $F = \sum_{j \in J_i} \lambda_j^{J_i} F_j$.

Note that the protocol works correctly if we assume the network is synchronous and that a *node* only fails at the beginning of the protocol. Basically what this protocol does is to agree on a set J_i in which index of *node*'s share to do a Lagrange interpolation from. One of the advantages of the protocol is that one node failures will not fail the whole protocol as long as the number of failures is less than predefined threshold.

If we allow arbitrary failures, then it might be the case that an agreed set J_i is different on each node. One way to handle that case is to have a consensus among all *node*s to agree on a set to do interpolation from. The author also proposed another way that is an extension of this initial protocol that is explained clearly in [30].

3 Research Methodology

Due to the amount of research application in the field of privacy-preserving data aggregation, our focus in this thesis is to propose such protocols that do not rely on the specific application domain. Instead, our proposed protocols are generic protocol in the sense that they could be applied to any different application domains. In this chapter, we explain the underlying network structure where the protocol can be applied on and the end goal of the protocols along with their assumptions, we also explain on how we evaluate our protocol.

3.1. Protocol Requirements

To fully answer all of our research question and its sub-questions, several requirements need to be achieved by our proposed protocol. Since our focus in this research is working in a general case of not only working in one specific application domain, we focus our requirement and protocol description in the term of *graph* network where there exist several *nodes* that are connected to each other. Requirements that need to be fulfilled for the protocols thus are:

- **Correct** For every node in the network that is included in the aggregation set, and in which all of them have their secret value, the protocol needs to be able to correctly compute the summation of all of those secret values at the end of the protocol run.

- **Privacy Preserving** In any steps of protocol run, a secret value of a node should only be known by the node that owns it. Note that this requirement does not limit the knowledge of aggregation result to a particular node or some nodes. Instead, the aggregation result of the protocol is regarded as public knowledge and may or may not be known by any particular user in the network. In this thesis, we focus on that the knowledge of aggregation result should be known by every node that contributes to the aggregation set.

- **Decentralized** There is no authority that has full control in the network. The nodes in the network should be able to act as an initiator of the protocol, and they should all conform to the protocol specification.

- **Direct Neighbors Knowledge** Every node in the network should know and be able to communicate (bidirectionally) with their direct neighbors. Total knowledge of the network in term of any other nodes and nodes connection beyond this direct neighbors is not required.

- **Dynamic Network Structure** The protocol should be able to handle changes in the network in term of addition or deletion of nodes, addition or deletion of connection, or node failure.

3.2. Network Structure

Let **G** denote a undirected graph network that consists of N number of *nodes* denoted as P_i where $i \in \{1,...N\}$. Each *node* in **G** may be connected to other *nodes* with a bidirectional edges connection. We do not assume any particular network structure in the protocol. Every *node* has a corresponding pair of secret and public key, and every node in the network have a knowledge of every other node's public key of Paillier [37]. This knowledge is used to provide additively homomorphism properties between any node. Each of P_i hold a secret value m_i that should only be known by P_i only. Every node also has an ability to compute a function $\mathcal{E}()$ and $\mathcal{D}()$, Paillier encryption and decryption function.

In a peer-to-peer network, any given node only knows the information of connectivity between itself and its direct neighbors. The overall structure of **G** may change from time to time, thus keeping the knowledge of the whole network will be time and space consuming. Any arbitrary *node* may be used to relay messages from any other *node* so that messages can be exchanged between any two *node*. For the sake of simplicity, in describing on how the protocol works in this section, we assume that such change in the network **G** may happen only before the start of "Initiate" phase. That is the protocol works correctly if no new node is added, removed, or failed during one run of the protocol. The protocol still works if the network structure in **G** changed and protocol gets to run again.

The protocol does not use external *aggregator* that resides outside the network. All aggregation calculation is done in the *node* within the network. Thus, we assume that every *node* in the network has capabilities to compute *additive homomorphic encryption* that is required in the protocol. We also assume that the underlying communication channel of the network is an authenticated channel, that is every node in the network can be sure that the messages they received are indeed originating from the node that sends them.

3.3. Protocol Goal

The goal of the protocol is to compute a summation of m_i of *some* of the *nodes*' secret value without leaking any information of the individual value to any other party, that is, at the end of the protocol the value of $\sum_j m_j$ where $j \in I \subseteq \{1,...,N\}$ will be known by P_j, we refer to this set as an aggregation set. We argue that computing the total of *some* of the *nodes*' secret value as opposed to all *nodes* is more practical in peer-to-peer network scenario. The definition of how we include the *node* membership in the aggregation set is defined in the protocol description.

The adversarial model used in the protocol is *honest but curious*, every party behave according to the protocol but may try to find out more information by reading every message that is being exchanged through them [48] [30]. We also assume that at least two of the *nodes* that contribute to the aggregation set are not colluding with each other, as it is trivial to conclude individual *node* secret value if every other *nodes* collude but one by subtracting the individual secret value of colluding *nodes* from the aggregated sum of protocol result [24].

3.4. Protocol Evaluation

Our research is aimed to provide privacy-preserving data aggregation in peer-to-peer network scenario. Therefore, we focus our evaluation on the correctness and privacy-preserving properties of our proposed protocols. To argue about these two properties, we also provide proof in the protocol description that they indeed conform with this requirements. Due to their nature of protocol in a peer-to-peer network, we also evaluate our proposed protocols in term of running time that they require in achieving the goal. To measure communication cost of the protocol instead of implementing and simulating the protocol in a simulated network environment, we measure the total message being sent to the network instead. Therefore our evaluation of protocol runtime excludes the time required if the protocols are implemented in real network scenario.

Due to lack of related research that specifically aims to solve similar goal as our work, we evaluate or proposed protocol using Protopeer [22] peer-to-peer framework modeling. The network being used to evaluate our protocols are generated randomly based on parameters that we explain in details in Chapter 6 using classes that are provided in ProtoPeer: `BootstrapServer`, `BootstrapClient`, and `NeighborManager` class. Since our protocols do not require a specific construction of network structure for them to function, network structure that is provided by ProtoPeer are sufficient.

4

Paillier-based Privacy-Preserving Data Aggregation Protocol

In this chapter, we propose our first protocol for privacy-preserving data aggregation in a peer-to-peer network. Our proposed protocol uses Paillier cryptosystem, utilizing its additive homomorphism property and secret sharing techniques to share noise values that help in achieving privacy. It also handles inherently unknown and dynamic network structures that exist in a peer-to-peer network by having a parameter to limit aggregation set generation in the nearest nodes only. We also give prove that our proposed protocol is correct and privacy-preserving in honest but curious adversarial model.

4.1. Protocol Description

Each node in the network has these following phase to achieve the protocol goal.

1. **Initiate** Any arbitrary *node P_i* in the network **G** may begin entering initiate step at any time instance as long as it has two or more neighbors connected to them.

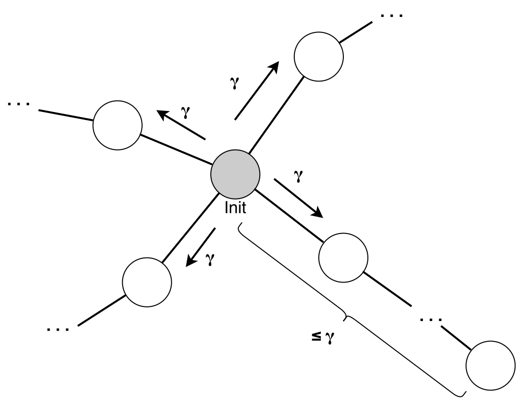

Figure 4.1: Because there is no assumption of the knowledge of the network, a node that initializes the protocol sends a γ parameter along with the initial message. At the end of the protocol, every node that has hop of less than or equal to γ are included in the aggregation set.

This *node* P_i decide on a parameter γ that define the maximum hops of neighboring *nodes*. Any other *nodes* within reach of γ hops or lower from initiating *node* are included in aggregation set.

Initiating *node* sends messages with type INITIATE and γ parameter to all of its direct neighbors. The message also contains information about i as a root of the initiator, that is the index of P_i. The node P_i then may enter "Receive Reply-Initiate" phase.

2. **Receive Initiate** Any *node* P_j that receive INITIATE message from P_i need to compute a noise value n_j so that

$$\sum_j n_j = 0 \tag{4.1}$$

with $j \in neighbor(P_i)$ and $neighbor(x)$ is a function that return the set of indices of x's direct neighbor, as ilustrated in Figure 4.2. To put it simply, P_j needs to agree with its direct siblings (by viewing P_i as a parent) of a value that if summed together will result in zero. The steps of how to generate and exchange such values is described deeply in the next section.

After agreeing on such value n_j the *node* check the γ parameter. If the γ value is more than one the node sends a message with a type AGGREGATE and decrease γ parameter by 1 to all of its direct neighbors except P_i. The message still contains i information as the root of the initiator. If γ value is not more than one, then the *node* does not send such message to its neighbors.

P_j then waits for all its direct neighbors to reply with REPLY-AGGREGATE message (if it sends any) or after a brief period of waiting, it ignores the neighbors that send no reply. Let P_k be such neighbors where k denote index of neighbors that reply with such message type along with an encrypted value received from P_k denoted as $\mathcal{E}_{root}(m'_k)$. P_j then computes

$$\mathcal{E}_{root}(m_j).\prod_k \mathcal{E}_{root}(m'_k).\mathcal{E}_{root}(n_j) \tag{4.2}$$

which under additive homomorphism will yield to

$$\mathcal{E}_{root}(m_j + \sum_k m'_k + n_j) \tag{4.3}$$

P_j sends a message with a type REPLY-INITIATE to P_i along with this value by denoting it as $\mathcal{E}_{root}(m'_j)$.

Notice that if node P_j doesn't receive any replies from its direct neighbors (or if it is not sending any message because of the γ parameter), then it sends this following value instead

$$\mathcal{E}_{root}(m'_j) = \mathcal{E}_{root}(m_j).\mathcal{E}_{root}(n_j) \tag{4.4}$$

$$= \mathcal{E}_{root}(m_j + n_j) \tag{4.5}$$

3. **Receive Aggregate** Any *node* P_k that receive AGGREGATE message from P_j check the corresponding γ parameter, if it has value more than 1 then it forwards AGGREGATE message further to their direct neighbors except P_j by first decreasing

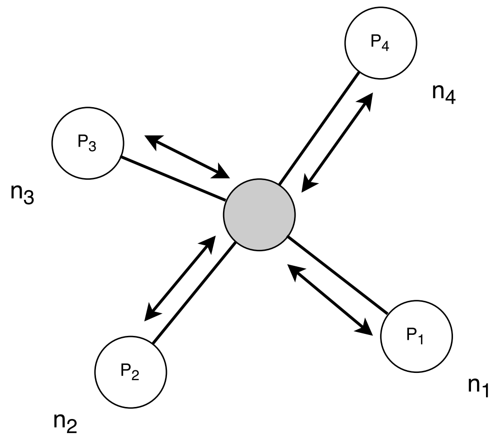

Figure 4.2: Nodes that are directly connected to the initiating node exchange values n_j with each other with a requirement that the values are summed to zero. When sending aggregation result in the next step, these nodes send the value along with the noise to ensure privacy from the initiating node.

γ parameter. It then waits for all of its direct neighbors except P_j to send a message with type REPLY-AGGREGATE or after a brief period of waiting ignores the neighbors that send no reply, thus treating that neighbors as a failure. After receiving REPLY-AGGREGATE messages P_k computes

$$\mathcal{E}_{root}(m_k).\prod_l \mathcal{E}_{root}(m_l) \tag{4.6}$$

where l denote the neighbors' index of P_k that give reply message after brief period of time and $\mathcal{E}_{root}(x)$ is an encryption of message x using Pailier under public key of root of the message initiator, that is the *node* P_i. Under an additive homomorphism, computing this value will yield to

$$\mathcal{E}_{root}(m_k + \sum_l m_l). \tag{4.7}$$

P_k sends a message with a type REPLY-AGGREGATE to P_j along with this value by denoting it as $\mathcal{E}_{root}(m'_k)$. An illustration can be seen in Figure 4.3.

If the γ value is not more than one (thus making it one of the leaves in the spanning tree), then P_k send a message with type REPLY-AGGREGATE to P_j along with $\mathcal{E}_{root}(m_k)$ value denoted as $\mathcal{E}_{root}(m'_k)$.

4. **Receive Reply-Initiate** Any node P_i that already do "Initiate" phase may enter this phase. P_i needs to wait for all of its direct neighbors P_j that receive INITIATE message in the previous steps to reply with REPLY-INITIATE along with $\mathcal{E}_{root}(m'_j)$ value. P_i then computes the total summation of aggregation set as follow

$$m_{sum} = m_i + \mathcal{D}_{root}(\prod_j \mathcal{E}_{root}(m'_j)), \tag{4.8}$$

where $\mathcal{D}_{root}(x)$ denote a Pailier decryption function of ciphertext x using P_i secret key (that is obviously known by P_i).

Node P_i then sends a message with type BROADCAST-SUM along with m_{sum} value and a γ parameter that is the same with the one used in "Initiate" phase.

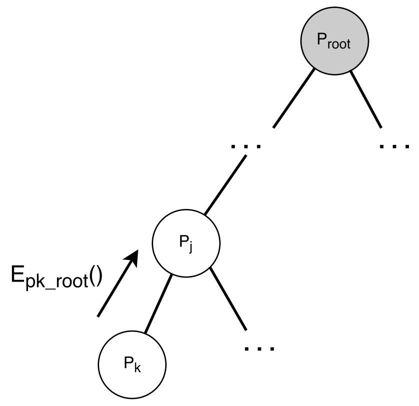

Figure 4.3: Nodes P_k that are not directly connected to the initiating node encrypt their secret value using the root's public key and send it to their parent P_j. In this case the privacy in ensured because P_j cannot see the secret value.

5. **Broadcast Sum** Any arbitrary node that receives a message with type BROADCAST-SUM store m_{sum} information as a total value in the aggregate set. The node also checks the γ parameter, if it has value more than one the node forward the BROADCAST-SUM message further to its direct neighbors except the one that sent it the message with γ parameter decreased by one.

4.2. Exchanging Noise Values

In "Receive Initiate" phase of protocol description we mention that the protocol requires nodes that are involved in those phases to exchange noise values with their direct siblings such that the total of those noise value will result in value zero. In this section, we explain on how to achieve that in detail.

Our approach slightly follows additive secret sharing exchange technique in [34] with modification. Let us consider P_j where $j \in neighbor(P_i)$ denotes sibling nodes that are directly connected to the initiating node P_i. In the "Initiate" phase of the protocol, initiating node P_i also sends to each of its direct neighbors the information about a $neighbor(P_i)$ so that every node that is directly connected to P_i have information about their direct siblings.

Let K denote the cardinality of set $neighbor(P_i)$, that is the number of P_i direct neighbor. Every node P_j create $K-1$ random shares integers $s_{j,k}$ with k corresponds to the index of their direct siblings, note that $s_{j,j}$ is not defined as this point. Assuming the existence of authenticated channel in the network, every P_j then could sends $s_{j,k}$ to their direct siblings P_k by relaying them through P_i.

After sending their own generated shares $s_{j,k}$ and receiving $M-1$ number of shares from other siblings, P_j then computes

$$s_{j,j} = -\sum_{k;k\neq j} s_{k,j}. \tag{4.9}$$

Every P_j now have a noise value that have summation value equal to zero as follow:

$$n_j = \sum_{k} s_{j,k}. \tag{4.10}$$

Proof: It is indeed $\sum_j n_j = 0$. Because by construction we have

$$\begin{aligned}
\sum_j n_j &= \sum_j \sum_k s_{j,k}, && (4.11)\\
&= \sum_j \sum_k s_{k,j}, && (4.12)\\
&= \sum_j \Big(\sum_{k;k\neq j} s_{k,j} + s_{j,j} \Big), && (4.13)\\
&= \sum_j \Big(\sum_{k;k\neq j} s_{k,j} - \sum_{k;k\neq j} s_{k,j} \Big), && (4.14)\\
&= \sum_j 0, && (4.15)\\
&= 0. && (4.16)
\end{aligned}$$

4.3. Analysis

In this section, we give an evaluation of our protocol proposal in term of a formal proof sketch of protocol correctness and privacy-preserving properties. Furthermore, we also evaluate our protocol computational complexity in the term of operation done in each sub-phases. Later on, we also evaluate the communication complexity of our protocol in term of a number of bits that each type of message in the protocol sends.

4.3.1. Correctness

We formally define correctness requirement as: For all nodes P_i that involves in the protocol run, if all of them are honest then the value of $m_{sum} = \sum_{i \in I} m_i$, where I is the aggregation set, is computed correctly and be known by all of them.

The protocol works by constructing a spanning tree from the network **G** with initiating node acting as tree's root, and the traversal of other nodes is done in breadth-first approach. Such spanning tree has a depth (that is, the number of edges connection between root and furthest leaf) of less than or equal to the γ parameter. Therefore we can see the γ parameter as a maximum level that the root is willing to do the traversal. In analyzing the correctness, we can see the protocol process in three different sub-phase: (1) initiating node and its direct neighbors, (2) internal (non-leaves) node, and (3) external (leaves) nodes.

Consider the first sub-phase of initiating node acting as the root of the spanning tree and its direct neighbors. Let P_i denote the initiating node, and P_j denote its direct neighbors. In "Receive Reply-Initiate" phase of the protocol, each P_j send m'_j to P_i. Let assume for now that m'_j contains the encryption of the original m_j value along with all values of its subtree (if we treat P_j as a root of this subtree) plus the noise value n_j. At the end of "Receive Reply-Initiate" phase, we indeed have m_{sum} the total sum of secret values in the aggregation set because,

$$\begin{aligned}
m_{sum} &= m_i + \mathcal{D}_{root}(\prod_j \mathcal{E}_{root}(m'_j)), && (4.17)\\
&= m_i + \mathcal{D}_{root}(\mathcal{E}_{root}(\sum_j m'_j)), && (4.18)\\
&= m_i + \sum_j m'_j, && (4.19)\\
&= m_i + \sum_j (m_j + sub(P_j)) + \sum_j n_j, && (4.20)\\
&= m_i + \sum_j (m_j + sub(P_j)) + 0, && (4.21)\\
&= \sum_{i \in I} m_i, && (4.22)
\end{aligned}$$

where $sub(x)$ denote the total value of the subtree of x node except x's secret value. If P_j is a leaf node, then this becomes trivial as $sub(P_j) = 0$. Because initiating node broadcast m_{sum} to all of other nodes that involves in aggregation set I, then it is the case that every one of them have m_{sum} information.

Previously, we assume the existence and correctness of encryption of m'_j. This is indeed a correct assumption due to our protocol construction. Consider the second sub-phase in internal nodes, they are the type of node that is not a leaf node and not a direct neighbor of initiating node. In "Receive-Initiate" phase of protocol description, encryption of m'_j is constructed as either

$$\begin{aligned}
\mathcal{E}_{root}(m'_j) &= \mathcal{E}_{root}(m_j).\mathcal{E}_{root}(n_j), && (4.23)\\
&= \mathcal{E}_{root}(m_j + n_j), && (4.24)
\end{aligned}$$

if P_j is a leaf node which make it trivial as already discussed in previous paragraph, or

$$\begin{aligned}
\mathcal{E}_{root}(m'_j) &= \mathcal{E}_{root}(m_j).\prod_k \mathcal{E}_{root}(m'_k).\mathcal{E}_{root}(n_j), && (4.25)\\
&= \mathcal{E}_{root}(m_j + \sum_k m'_k + n_j), && (4.26)
\end{aligned}$$

if P_j is not a leaf node. If once again we assume an existence and correctness of m'_k, both of them fulfill our previous assumption that m'_j contains the encryption of m_j and all of the node's subtree.

Our assumption about m'_k is indeed also a correct one. As m'_k is constructed to have exactly such property. Described in "Receive Aggregate" phase of the protocol description, m'_k is constructed as either

$$\begin{aligned}
\mathcal{E}_{root}(m'_k) &= \mathcal{E}_{root}(m_k).\prod_l \mathcal{E}_{root}(m_l), && (4.27)\\
&= \mathcal{E}_{root}(m_k + \sum_l m_l), && (4.28)
\end{aligned}$$

for non-leaf node P_k with one or more child nodes P_l. Or as

$$\mathcal{E}_{root}(m'_k) = \mathcal{E}_{root}(m_k), \qquad (4.29)$$

for leaf node P_k. Thus, fulfilling our correctness assumption about it.

4.3.2. Privacy Preserving

We formally define privacy-preserving requirement as: For all nodes P_i that involves in the protocol run, if all of them are honest then the value of m_i is known only by P_i, where $i \in I$ and I is the aggregation set.

All encryption in the protocol is done using Pailier cryptosystem with the public key of initiating node. Because of that, it is self-evident that any node other than initiating node cannot decrypt the encryption of secret value from any other node involving in the aggregation set. We are now left to focus on our privacy preserving analysis in the initiating node.

We argue that the initiating node, although holds the private key required to decrypt any ciphertext in the protocol, have no ability to see or deduce the actual secret value of any other node involved in the protocol. In our protocol description, we require that arbitrary node can only initiate the protocol if it is connected to at least two of other nodes, this has its own reasoning. If a node is connected to zero other nodes, then that node does not need to run any protocol to compute aggregation value as the value is its own secret value. If a node is connected to just one other node, then the aggregation value can be computed, but it is impossible to achieve privacy because it is trivial to calculate other node's secret value by subtracting the aggregation result with its secret value. Thus, at least two of connection to other nodes is required.

Privacy of nodes that are not directly connected to the initiating node is guaranteed because when they send encryption of their secret value, the value is sent through other node and multiplied with the intermediate node's secret value. Thus not only it calculates the addition of secret value in a homomorphic manner, but it also prevents the initiating node to recover individual value because it only receives the encryption of addition results only.

Nodes P_j that are connected directly to the initiating node P_i need another way to ensure their privacy, that is why we require these nodes to exchange noise values n_j with a requirement that $\sum_j n_j = 0$, with their siblings. If these nodes have other nodes P_k connected to them (thus acting as their children in spanning tree), then the privacy already achieved because when they send encryption of m'_j to the initiating node it contains encryption of addition result of m_j and the node's subtree secret value. This noise value is useful particularly in the case of when P_j is one of the leaves in the spanning tree. Their secret value's privacy is guaranteed because when P_i only receive individual encryption of m'_j from P_j it contains n_j, thus disabling the decryption to individual secret value. Decryption process only works correctly in recovering them only if P_i receive values from other P_j as well, in the form of total aggregation only.

4.3.3. Computational Complexity

In analyzing the computational complexity of the Paillier-based protocol, we list the number of operations performed by nodes per execution of their respective sub-phases of the protocol and also the number of such sub-phases being invoked in the network. The number of operations performed depends on several variables that are listed in Table 4.1.

Table 4.2 lists the number of invocation of protocol's sub-phases in the network. The table also lists the number of operation being executed per invocation of a subphase. Note that depending on their role in one run of the protocol, each node exe-

Symbol	**Description**
$\mathbf{G}$	The network where the protocol run
N	Number of nodes in the network
$\Delta(\mathbf{G})$	Maximum degree of the network
γ	Maximum depth of the protocol run, relative to the initiating node
I	The aggregation set
t	Bit-length of secret value
α	Bit-length of noise value
β	Bit-length of ciphertext
h	Bit-length of common header (message type flag, gamma, root identifier, sender identifier, receiver identifier)

Table 4.1: Symbols used in computational and communication analysis of Paillier-based privacy preserving data aggregation protocol

Sub-phase	**Occurrence**	Operation per Occurrence		
		Encryption	**Decryption**	**Multiplication**
Initiate	1		1	$O(\Delta(\mathbf{G}))$
Noise exchange	$O(\Delta(\mathbf{G}))$			
Receive Initiate	$O(\Delta(\mathbf{G}))$	1		
Receive Aggregate	$O(\Delta(\mathbf{G})^{\gamma})$	1		
Reply-Initiate	$O(\Delta(\mathbf{G}))$			
Receive Reply-Initiate	$O(\Delta(\mathbf{G}))$			
Receive Reply-Aggregate	$O(\Delta(\mathbf{G})^{\gamma})$			$O(\Delta(\mathbf{G}))$
Broadcast Sum	$\|I\|$			

Table 4.2: Computational complexity analysis of the first protocol

cutes different sub-phases in the protocol with others and may invoke several of such sub-phases. The overall computational cost of the protocol is then can be obtained by multiplying values in each row. Notice that in giving the computational complexity analysis we use worst-case scenario notation as the actual computation of the protocol depends heavily on the network structure that it runs on. Also, the analysis in here excludes the key generation phase and node initialization that are done by every node in the network, since the phase only executed one time by each node and not executed in each invocation of the protocol.

The overall protocol complexity is dominated by aggregation sub-phases (the rec eive-aggregate and receive reply-aggregate) in an exponential factor of the γ parameter. This analysis comes from the upper bound of a number of node in a full tree with the degree equal to $\Delta(\mathbf{G})$ and height equal to γ [44]. The multiplication operations are executed in β-bit number of ciphertexts. Notice that every node in the network at most executes multiplication of β-bit numbers $\Delta(\mathbf{G})$ times. Therefore, the operations of the protocol are spread to the node in the network that is included in the aggregation set.

Message type	**Bit-length**
INITIATE	$h + O(\Delta(\mathbf{G}))$
NOISE-EXCHANGE	$h + \alpha$
REPLY-INITIATE	$h + \beta$
AGGREGATE	h
REPLY-AGGREGATE	$h + \beta$
BROADCAST-SUM	$h + t$

Table 4.3: Communication complexity analysis of the first protocol

4.3.4. Communication Complexity

We analyze the communication complexity of the Paillier-based protocol by listing the number of bits transmitted by each type of messages in the network and also the number of such message type being transmitted in one protocol run. Table 4.1 lists variables that we use in giving this analysis.

Table 4.3 lists the number of bits transmitted by each type of messages in the network in one message transmission. The total message in a bit of the network can roughly be obtained by multiplying values in Table 4.3 with the number of occurrence per sub-phase in Table 4.2. All of the message share common header field information, with bit-length, denoted as h, that contains information about message type, gamma parameter, an identifier of the root, identifier of the sender, identifier of the receiver. The actual length of this header field depends on the underlying implementation.

Notice that the INITIATE message type also sends $O(\Delta(\mathbf{G}))$ bits of data along with the common header information. This information contains the identifier of nodes that are directly connected to the initiating node and is used in the noise exchange phase of the protocol. Meanwhile, REPLY-INITIATE and REPLY-AGGREGATE message type send roughly the same message length in bits as their role are similar, that is to send aggregation result upstream to the direction of the initiating node.

4.4. Achieving Overall Network Aggregation

The protocol constructs an aggregation set by generating a spanning tree with the initiating node as the root of the tree. The depth of the traversal of the tree depends on γ parameters. The protocol finds nodes with up to γ length from the initiating node. If we let γ more than or equal to the number of hops of the farthest node from the initiating node than it is easy to show that the protocol will have all nodes in the network **G** in the aggregation set.

To have this information on the number of hops required to reach the farthest nodes from any other nodes, we can have another step when a new node is connected to any node that is already within the network. Let every node in the network have a variable γ_{max} that has a value of the number of hops required to reach the farthest node. Upon receiving a new connection in the network, a node P_i compare its current γ_{max} with the value one, if it is more than one than P_i sets its γ_{max} to one. P_i then broadcasts to all of its direct neighbor a message containing an information of $\gamma_{new} = 2$. Any nodes that receive information of γ_{new} message compare their current γ_{max} with this value, updating it if the value is less than γ_{new}. Then they further broadcast a new γ_{new} incremented by one message to their other direct neighbor, and every network recursively does the same until every node receives this message, having their respective γ_{max} information updated.

5 Secret Share-based Privacy-Preserving Data Aggregation Protocol

In this chapter, we propose another protocol for privacy-preserving data aggregation in a peer-to-peer network. The second protocol only utilizes additive secret sharing technique without relying on Paillier cryptosystem, unlike the first protocol. Similar to the first protocol, the second protocol does not require an overall network structure for it to achieve its task. Instead, it also utilizes a threshold parameter to limit aggregation set membership to the nearest nodes only.

5.1. Protocol Description

Our first proposed protocol uses both Paillier cryptosystem for its additive homomorphism properties and secret sharing to generate noise values between initiating node's children. Here we propose another alternative protocol that only uses secret sharing entirely without having to rely on Paillier cryptosystem for its additive homomorphism. We illustrate the protocol in Figure 5.1. This alternative protocol is useful where there is a limitation on having a public knowledge of all of the public key in the network and the limitation in the nodes when having Paillier cryptosystem is not possible. However this protocol imposes another limitation that is required to be fulfilled by the network, namely that upon constructing a spanning tree of the aggregation set as defined in our previous protocol any internal nodes need to be connected to at least three direct nodes (one is the parent upstream node and two other is the children downstream node). The initiating node can still be connected to two nodes only, the children. This other requirement is required to ensure privacy in all nodes of the network. In defining our alternative protocol we assume that all internal non-root nodes in the spanning tree are connected to at least three other nodes, one is the upstream parent node and at least two downstream children nodes. The protocol works as follow:

1. **Pre-Initiate** Any arbitrary *node* P_i in the network **G** may begin entering initiate step at any time instance as long as it has two or more neighbors connected to them. Before doing so, they execute this phase to query all of their direct

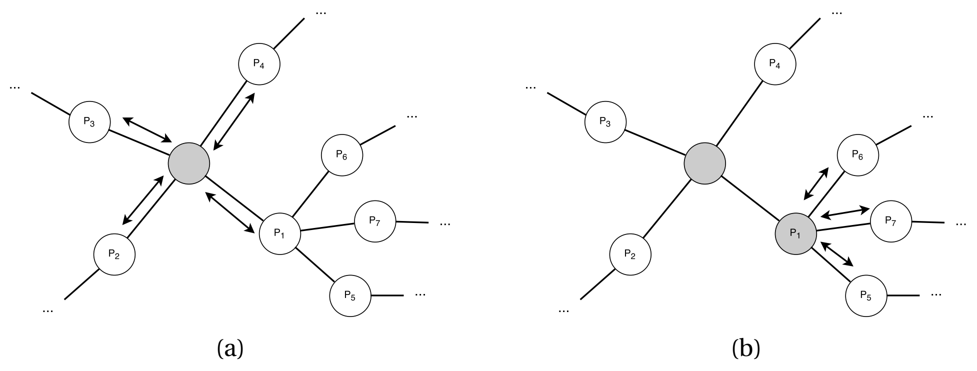

Figure 5.1: Our second proposed protocol works as follow: in (a) initiating node does its initiate phase and then each node recursively execute their own initiate phase to their children as depicted in example of (b).

neighbors' status. The node P_i only includes its direct neighbors if they are active and not already in the aggregation set.

2. **Initiate** This *node* P_i decide on a parameter γ that define the maximum hops of neighboring *nodes*. Any other *nodes* within reach of γ hops or lower from initiating *node* are included in aggregation set.

   Initiating *node* sends message with a type INITIATE and γ parameter to all of its direct neighbors that are included in the "Pre-Initiate" phase. The message also contains information about i as a root of the initiator, that is the index of P_i. The node P_i then may enter "Receive Reply-Initiate" phase.

3. **Receive Initiate** Any *node* P_j that receive INITIATE message from P_i need to compute a noise value n_j so that

$$\sum_j n_j = 0 \tag{5.1}$$

   with $j \in neighbor(P_i)$ and $neighbor(x)$ is a function that return the set of indices of x's direct neighbor. To put it simply, P_j needs to agree with its direct siblings (by viewing P_i as a parent) of a value that if summed together will result in zero. The steps of how to generate and exchange such values is described deeply in Section 4.2.

   After agreeing on such value n_j the *node* check γ parameter, if its value is more than one the node executes its own INITIATE phase by first doing PRE-INITIATE with γ parameter decreased by 1, in doing so the node P_j also ignore the node P_i in determining its direct neighbors. The message still contains P_i information as the original initiator. If γ value is not more than one, then the *node* does not send such message to its neighbors, instead it sends a message with type REPLY-INITIATE along with $m'_j = m_j + n_j$ to P_i.

4. **Receive Reply-Initiate** Any node P_i that already do "Initiate" phase may enter this phase. P_i needs to wait for all of its direct neighbors P_j that receive INITIATE message in the previous steps to reply with REPLY-INITIATE along with m'_j value. P_i then computes the total summation of aggregation set as follow

$$m_{sum} = m_i + \sum_j m'_j. \tag{5.2}$$

Node P_i then sends a message with type BROADCAST-SUM if and only if it is the original initiator (this can be checked by the message that contains original i index) along with m_{sum} value and γ parameter that are the same with the one used in "Initiate" phase.

If this node is not the original initiator, it sends $m_{sum} + n_j$ to its parent instead with the message type REPLY-INITIATE, note that the parent treats the value as m'_j.

5. **Broadcast Sum** Any arbitrary node that receives a message with type BROADCAST-SUM store m_{sum} information as a total value in the aggregate set. The node also checks the γ parameter, if it has value more than one the node forward the BROADCAST-SUM message further to its direct neighbors except the one that sent it the message with γ parameter decreased by one.

5.2. Analysis

In this section, we give an evaluation of our protocol proposal in term of a formal proof sketch of protocol correctness and privacy-preserving properties. Furthermore, we also evaluate our protocol computational complexity in the term of operation done in each sub-phases. Later on, we also evaluate the communication complexity of our protocol in term of the number of bits that each type of message in the protocol sends.

5.2.1. Correctness

We formally define correctness requirement as: For all nodes P_i that involves in the protocol run, if all of them are honest then the value of $m_{sum} = \sum_{i \in I} m_i$, where I is the aggregation set, is computed correctly and be known by all of them.

Similar to the first protocol, the second protocol in its core works by constructing spanning tree of the network **G** with the initiating node acting as the root of the tree. The spanning tree that is constructed has a depth (the number of edges connection between root and furthest leaf) less than or equal to the γ parameter. The traversal is done in breadth-first fashion. Unlike the first protocol, however, the second protocol has only one core phase that is executed recursively, the noise exchange using secret sharing phase.

Let us consider the initiating node P_i that acts as the root of the spanning tree in the network **G**. Recall that we assume that all internal non-root nodes in such spanning tree are connected to three other nodes (one upstream parent nod and two other children). Let P_j denote the nodes that are directly connected to P_i, all of them have their own aggregation result m'_j of their own sub-tree (such sub-tree that they become the root of it) and noise value n_j. In the "Receive Reply-Initiate" phase, the final summation is indeed computed correctly because

$$
\begin{aligned}
m_{sum} &= m_i + \sum_j m'_j, && (5.3)\\
&= m_i + \sum_j (m_j + sub(P_j)) + \sum_j n_j, && (5.4)\\
&= m_i + \sum_j (m_j + sub(P_j)) + 0, && (5.5)\\
&= \sum_{i \in I} m_i. && (5.6)
\end{aligned}
$$

where $sub(P_j)$ denotes the total value the children of sub-tree where P_j is the root. Since the protocol recursively execute this steps throughout the spanning tree, m'_j is constructed correctly in similar approach as the construction of m_{sum}.

5.2.2. Privacy Preserving

We formally define privacy-preserving requirement as: For all nodes P_i that involves in the protocol run, if all of them are honest then the value of m_i is known only by P_i, where $i \in I$ and I is the aggregation set.

We argue that even without using additively homomorphism properties of Paillier as in first protocol, the second protocol indeed also achieve privacy-preserving properties. Recall that our assumption requires any internal non-root nodes to be connected to at least three other nodes where two of them are the children. This property is required to ensure privacy preserving requirement that comes from noise exchange steps.

In every phase of the protocol run, every node P_i in the network only have knowledge of their own secret value m_i, or if they are in the process of aggregating data (such as in "Receive Reply-Initiate" phase) an information of their children secret value (or the total of their sub-tree value) m'_j added with the noise value n_j. The original value can only be recovered if the node already obtains all of its children messages so that the noise values can be canceled out (recall that $\sum_j n_j = 0$).

5.2.3. Computational Complexity

In analyzing the computational complexity of the Secret share-based protocol, we list the number of operations performed by nodes per execution of their respective sub-phases of the protocol and also the number of such sub-phases being invoked in the network. The number of operations performed depends on several variables that are listed in Table 5.1.

Table 5.2 lists the number of invocation of protocol's sub-phases in the network. The table also lists the number of operation being executed per invocation of a sub-phase. Note that depending on their role in one run of the protocol, each node executes different sub-phases in the protocol with others and may invoke several of such sub-phases. The overall computational cost of the protocol is then can be obtained by multiplying values in each row. Notice that in giving the computational complexity analysis we use worst-case scenario notation as the actual computation of the protocol depends heavily on the network structure that it runs on.

Unlike the first protocol, the secret share-based protocol relies fully on the noise that is added to the secret value. The initiating phase and also noise exchange phase

Symbol	**Description**
$\mathbf{G}$	The network where the protocol run
N	Number of nodes in the network
$\Delta(\mathbf{G})$	Maximum degree of the network
γ	Maximum depth of the protocol run, relative to the initiating node
I	The aggregation set
t	Bit-length of secret value
α	Bit-length of noise value
h	Bit-length of common header (message type flag, gamma, root identifier, sender identifier, receiver identifier)

Table 5.1: Symbols used in computational and communication analysis of Secret share-based privacy preserving data aggregation protocol

Sub-phase	**Occurrence**	Operation per Occurrence
		Addition
Pre-Initiate	$O(\Delta(\mathbf{G})^{\gamma})$	
Initiate	$O(\Delta(\mathbf{G})^{\gamma})$	$O(\Delta(\mathbf{G}))$
Noise exchange	$O(\Delta(\mathbf{G})^{\gamma})$	
Receive Initiate	$O(\Delta(\mathbf{G})^{\gamma})$	1
Receive Reply-Initiate	$O(\Delta(\mathbf{G})^{\gamma})$	
Broadcast Sum	$\|I\|$	

Table 5.2: Computational complexity analysis of the second protocol

are invoked recursively in the network up to the γ parameter. Every node in the network at most executes addition of α-bit numbers $\Delta(\mathbf{G})$ times. Similar to the first protocol, the operations are spread to the node in the network that is included in the aggregation set.

5.2.4. Communication Complexity

We analyze the communication complexity of the Secret share-based protocol by listing the number of bits transmitted by each type of messages in the network and also the number of such message type being transmitted in one protocol run. Table 5.1 lists variables that we use in giving this analysis.

Table 5.3 lists the number of bits transmitted by each type of messages in the network in one message transmission. The total message in bit of the network can roughly be obtained by multiplying values in Table 5.3 with the number of occurrence per sub-phase in Table 5.2. All of the message share common header field information, with bit-length, denoted as h, that contains information about message type, gamma parameter, identifier of root, identifier of sender, identifier of receiver. The actual length of this header field depends on the underlying implementation.

Notice that the INITIATE message type also sends $O(\Delta(\mathbf{G}))$ bits of data along with the common header information. This information contains the identifier of nodes

Message type	**Bit-length**
PRE-INITIATE	h
REPLY-PRE-INITIATE	h
INITIATE	$h + O(\Delta(\mathbf{G}))$
NOISE-EXCHANGE	$h + \alpha$
REPLY-INITIATE	$h + \alpha$
BROADCAST-SUM	$h + t$

Table 5.3: Communication complexity analysis of the second protocol

that are directly connected to the initiating node and is used in the noise exchange phase of the protocol. In REPLY-INITIATE message type, the protocol sends messages with bit-length equal to the header added by the bit-length of the noise. This is due to the fact that the second protocol is not using encryption and relies fully on the added noise.

6 Implementation and Experimental Result

We present our implementation and experimental result of our proposed protocols in this chapter. Both of the protocols, the first one using Paillier homomorphic cryptosystem and the second alternative protocol using only secret sharing techniques are presented here. We further discuss the differences between those two based on the experimental result. The implementation of the protocols is done in Java programming language on an Intel Core i7-4500U @ 1.80 GHz with 16GB of RAM machine.

6.1. Peer-to-Peer Framework Used

In implementing our protocol we utilize ProtoPeer [22], which further have been extended by DIAS [38], framework. The framework suits our intention well because it allows us to implement our protocol programmatically without having to dive too deep into the underlying network and message passing mechanism. It also allows us to not only build our protocol but also allows us to evaluate them, by using pre-existing sets of APIs. Using ProtoPeer also allows a further work to transition our implementation to live network with less time-consuming work.

In ProtoPeer and DIAS' context, every node in the network acts as an individual Peer that has its own self-contained application code definition called a Peerlet. Every interaction is modeled as either (1) message driven, when a node receives a message and depending on the type of the message act upon it, or (2) time driven when a timer that already set-up previously trigger an event on a certain time.

On the other hand, both of our proposed protocol is designed without assuming total knowledge of the network. We define the design of our protocols in the term of the behaviour of a node (or Peer, in ProtoPeer terminology) and what it should do when it receives a certain message type. As can be seen in Figure 6.1, this is achievable in the implementation of ProtoPeer context by first defining the message and then handling the behavior of each node based on the message received by them.

We generate our network structure using ProtoPeer's *BootstrapServer, Bootstrap-Client,* and *NeighborManager* peerlet classes. By providing a parameter of how many nodes that we want to generate in the network, these peerlet classes generate them and make connections between the nodes in the network. We do not create our own

```
public void handleIncomingMessage(Message message) {
        if(message instanceof PaillierMessage) {
            PaillierMessage paillierMessage =
                (PaillierMessage) message;
            switch (paillierMessage.messageType) {
                case INITIATE: {
                    ...
                }
                case NOISE_EXCHANGE: {
                    ...
                }
                case REPLY_INITIATE: {
                    ...
                }
                case AGGREGATE: {
                    ...
                }
                case REPLY_AGGREGATE: {
                    ...
                }
                case BROADCAST_SUM: {
                    ...
                }
            }
        }
    }
```

Figure 6.1: In protocol implementation we handle the behaviour by differentiating the type of message received

pre-defined network as our proposed protocols could handle arbitrary structure of the network and creating pre-defined network would fall outside the scope of our research interest.

Our experiments are run in *SimulatedExperiment* context of ProtoPeer. What it does is simulating network communication by not using the actual network infrastructure. Because of that, our experiments do not include the communication cost of the actual network communication as we have zero delays and zero loss experiment. To measure our network messaging capabilities, we measure the number of messages that are being sent and received in the network instead.

In the first protocol, the Paillier-based, we modified a Paillier protocol implementation from [28], adding capabilities to compute encryption from different object instances using a public key of the target.

6.2. Paillier-based Privacy-Preserving Data Aggregation

We experimented with four different scenarios that differ in the number of nodes or peers in the network: 500, 1000, 2000, and 5000 nodes. Results are shown in Table 6.1.

NUM_PEERS	Runtime (ms)	Total Messages
500	9,595	25,567
1,000	20,894	67,563
2,000	46,521	162,027
5,000	133,643	484,035

Table 6.1: Experiment result of Paillier-based protocol

We set the γ parameter, the maximum hops of the protocol, of all of them to be 10. Furthermore, we measured the total runtime of one aggregation protocol, and we also measured the total messages being sent/received in the whole network. In measuring the runtime of the protocol, we do not include the time it takes for all nodes to initialize its internal value, such as keypair and secret value generation. A total runtime is defined as the total time it takes (in ms) from the first INITIATE message until the last BROADCAST-SUM message, that is until all of the nodes included in aggregation set have a knowledge of the total sum result. We repeat our experiment ten times and taking the average result in measuring the total runtime.

Every node has a randomly generated 32-bits secret value number in $\mathbb{Z}_n$. Each of them also has their own keypair of Pailler with the length of 1024-bits. Recall that due to Paillier, our ciphertexts are in $\mathbb{Z}_{n^2}$. In the first phase of the protocol, we require an exchange of noises among all nodes that are directly connected to the initiating node. The information of ciphertext added with noise value is theoretically secure. However, we need to mask the secret value distribution. To mask the distribution of the secret values, we set the length of the noise to be a 128-bits number [33] [34]. To validate the correctness of our implementation, we also store every secret value that contributes in the aggregation set to an external storage (we implemented this in a static variable in Java) and by the end of the protocol we compare the sum of the value stored with the result of our protocol runs.

6.3. Secret Share-based Privacy-Preserving Data Aggregation

We used a similar approach in our second sets of the experiment of the second protocol as we did with the previous set for the first protocol. Result can be seen on Table 6.2. The only difference is that in the second protocol each node does not have to generate a keypair of Paillier. We use a similar definition of the total runtime as we did in previous protocol, it is measured as a total time it takes from the first PRE-INITIATE message until the last BROADCAST-SUM message.

The parameter of number length is also similar to the previous protocol experiment. We generated a random 32-bits secret value number, and we also use a 128-bits number for our values in the noise exchange phase. Correctness is also still validated by means of comparing the value from external storage with the resulting value of protocol runs.

6.4. Protocol comparison

As we can see from the experiment result from our first protocol, the Pailer-based requires approximately 40 percent more runtime compared with the second protocol,

NUM_PEERS	Runtime (ms)	Total Messages
500	4,413	28,677
1,000	11,001	76,671
2,000	29,390	191,215
5,000	102,228	585,583

Table 6.2: Experiment result of Secret share-based protocol

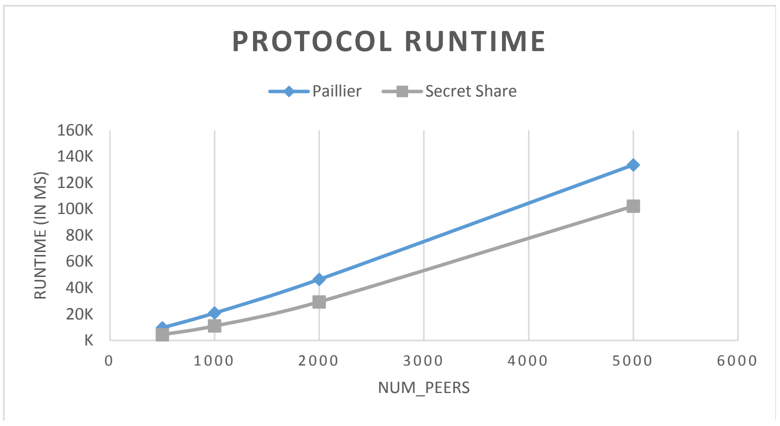

Figure 6.2: A comparison plot of protocol total runtime in two protocols.

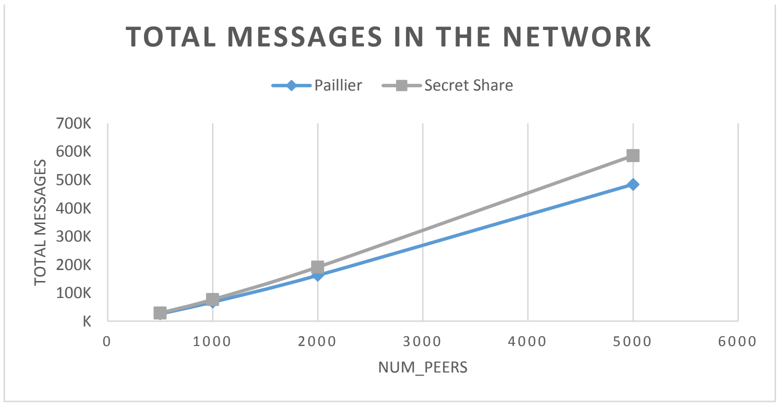

Figure 6.3: A comparison plot of total messages in the network of two protocols.

the secret-share based one, based on four number of peers scenario that we conducted in the experiment. This is to be expected because in the first protocol we require the multiplication of big number as part of additive homomorphism properties in each intermediate nodes of the network. In the second protocol, we only add the secret value that already added by noise value in order to aggregate the data. This requires fewer operation than what it requires in the first protocol.

However, based on our experiment in measuring total messages in the network, our first protocol sends approximately 16 percent fewer messages in the network compared with the second protocol, based on four number of peers scenario. This is due to the fact that in our second protocol we require each node to recursively invoke noise exchange phase with their direct siblings prior to sending secret values to their parent node. This mechanism does not exist in the first protocol as noise exchange phase is only required to be executed by the nodes that are directly connected to the initiating node.

7 Discussion and Future Work

Data is inherently valuable due to its capability when we could operate on it. Treating data as a secret while at the same time hindering its usage devalued it completely. On the other hand, blindly trust other parties to handle our private data is also not the best approach as they can misuse our data. Having a mechanism to provide operational capabilities while still maintaining the privacy of our is a desirable approach. We explore research in the field of power-grid and smart meter, wireless sensor network, crowdsensing and find out shared similarities among them in the sense that they all could be described as the protocol in the distributed network concept although they all differ in the field of application.

Protocols for privacy-preserving data aggregation have been proposed in earlier works. However, all of them assume a total knowledge of the network structure. Earlier works also assume roles of the node in the network. The nodes could be regarded as either normal node that only provides data and aggregator node that aggregate the data provided. Our research focuses on a less assumptive scenario where each node in the network only knows and could communicate with nodes that are directly connected to it. Our work also aims to solve a more general case where we want to compute some of the node in the network based on a parameter as opposed to computing all of the nodes in the network. One of the real world application of this scenario is in the field of decentralized power-grid where power is generated near where they are needed as opposed to one single power provider. Our focus in this research could be described by this following research question:

> *How can privacy be achieved in data aggregation of peer-to-peer network scenario where there is no authority and each node in the network have no knowledge of the overall network structure?*

This chapter's purpose is to revisit the research question and discuss our two proposed protocols of how they achieve the research goal. Later on, we also give research idea for the future work.

7.1. Discussion

We proposed two protocols of privacy-preserving data aggregation in a peer-to-peer network. The first protocol relies on additive homomorphism properties of Paillier cryptosystem and secret sharing scheme for noise exchange in nodes that are directly connected with initiating node. The second protocol relies solely on secret sharing scheme that executed recursively throughout the network, while requires more assumption in the network structure. Both of proposed protocols require no knowledge of overall network structure, and they both achieve privacy-preserving data aggregation in some of the node in the network based on hop threshold parameter.

Our proposed protocols differs with related works that mentioned in Chapter 2 because while the works mentioned there specifically solve privacy-preserving data aggregation in their own specific application domain, we choose to base own work on more abstract concept so that they can be applied in any domain (with small update), due to this reason we based our proposed protocol to work in peer-to-peer network scenario. Our proposed works also require no knowledge of overall network structure as storing this kind of complete information is space consuming, instead, every node in the network is required only to be able to communicate bi-directionally with their direct neighbors, and data aggregation is achieved in some of the network based on a hop parameter. Moreover, our protocols also require no separation of roles among the nodes, all of them could initiate the protocol and thus become the central of aggregation as long as it fulfills the requirements that are mentioned in Chapter 4 and Chapter 5.

Looking back into Section 1.6, we identify five sub-questions that motivate us in doing this research. The first question focus on what mechanism could be used in order to maintain the privacy of secret data of each node in the network. To answer this, our proposed protocols use a different mechanism to achieve privacy-preserving properties. The first protocol that we propose relies on additive homomorphism properties of Paillier cryptosystem while the second protocol relies solely on secret sharing mechanism that is recursively invoked.

The second sub-question focus on the dynamic structure of the network where nodes can be added or deleted be handled. Both of our proposed protocol handle this scenario well and correctly compute the aggregation result if nodes are not added or deleted during one protocol run. Since in the real world scenario we can safely assume that such event of nodes addition and removal are rare, then our proposed protocol would work most of the time correctly.

The third sub-question focus on the way the protocol handle lack of knowledge of the overall network structure. Both of our protocols solve this issue by having a γ parameter that bound the maximum number of hops the protocol should invoke the aggregation protocol sub-phase. By using this approach, a total network aggregation is still possible if the node that invokes initiate phase have an information of the hop count to reach the furthest node. This approach of storing only the hop of the furthest node is preferable storage-wise than storing an information of a whole network structure.

In the fourth sub-question, we discuss the mechanism to achieve privacy-preserving data aggregation without the availability of central authority. In both of our proposed protocol, we solve this issue by allowing every node in the network to spontaneously invoke the initiating protocol as long as it fulfills requirements mentioned in Chapter 4 and Chapter 5. By invoking the "Initiate" phase, we construct a spanning tree of the network with the maximum depth of γ and the initiating node acting as the root of this spanning tree.

Finally, in the fifth sub-question, we explore the suitable cryptographic primitive that could be used in solving privacy-preserving data aggregation in peer-to-peer network scenario. Our first protocol fully utilizes both additive homomorphism properties of Paillier cryptosystem along with secret sharing to achieve privacy. On the other hand, the second protocol only relies on secret sharing while requiring more assumption in the network structure. Both of the protocol are upper bounded by γ parameter exponentially.

7.2. Future Work

To the best of our knowledge, our proposed protocols are the first protocol to achieve privacy-preserving data aggregation in a peer-to-peer network where each node in the network does not need to store the overall network structure. Both of them, Pailier-based and secret share-based achieve both correctness and privacy-preserving requirement. However, we realize that there are rooms for improvement in these protocols.

- **Malicious security model.** In defining our two protocols description, we use *honest but curious* security model. That is, every party assumed to follow the protocol description while still be able to infer as much information as they want from received data. This approach is suitable to use in defining new protocols, as we already did. However, in a real-world scenario, there might be a case where some of the parties in the network behave maliciously and try to corrupt the network by not following the protocol definition. One of the examples is that a node in the protocol might act like it is connected to non-existence nodes in order to lie to another node that it has several direct neighbors. This is malicious due to the fact that the privacy of both of our protocols is guaranteed by noise exchange mechanism of nodes that are directly connected to the network. Acting malicious in this phase may thus break privacy. Future works should have this in mind and design according to the malicious security model.

- **Real world network testing.** In evaluating both of our protocols, we use simulated scenario of Protopeer's *SimulatedExperiment* context. This experiment context simulates network communication and also simulates nodes environment in one single machine. Furthermore we also generate our network structure using Protopeer's *BootstrapServer*, *BootstrapClient*, and *NeighborManager* classes without full control of the underlying network being generated. Future research topic may use real-world network data to evaluate the performance of the protocols. Testing in a real network environment where we have delay and loss in message transmission also an interesting idea to propose.

- **Application in specific field.** Our contribution of this thesis focus on protocols that work on peer-to-peer network scenario, it is a lot more abstract than related works that we reviewed where they focus on one specific application field. Applying our protocols in one specific field, for example testing them in decentralized power-grid where privacy is an important aspect to have, is one of an interesting research topic that could be proposed.

7.3. Final Remarks

Two of our proposed protocols achieve their intended goal to have privacy-preserving data aggregation in peer-to-peer network settings. To be precise, our proposal differs from related works that have been conducted previously in term of underlying assumption of the network structure. Our two proposed protocols require no knowledge of overall network structure that they run under. By only having abilities to communicate with their direct neighbors, each node in the network which our protocol run under could achieve data aggregation in the privacy-preserving way of several nodes based on a parameter provided.

In worse case scenario, both of our protocol complexity is exponential in the term of the γ parameter. However, due to their nature of being peer-to-peer protocol, every node in the network has a pretty low overhead in their computation. In the first protocol, every node in the network has computation that is upper bounded by multiplication operation in ciphertext domain in the maximum degree of the network only. While in the second protocol, the operation is conducted as an addition of noise value domain.

A

Appendix

A.1. Pailler-based run log example

```
#1
==DEBUG result is (500) 1075725634667
==protocol result is 1075725634667
++ Experiment Paillier based with NUM_PEERS = 500
== Total timerun 9477.0
== PEER INIT (incl. Paillier keygen)
sum 30881.0
avg 61.762
== Message count
INITIATE_MSG 20
NOISE_EXCHANGE_MSG 760
REPLY_INITIATE_MSG 20
AGGREGATE_MSG 8249
REPLY_AGGREGATE_MSG 8249
BROADCAST_SUM_MSG 8269
Total messages = 25567
```

A.2. Secret share-based run log example

```
#1
== mSum = 937476874671 DEBUG =937476874671
++ Experiment Secret share based with NUM_PEERS = 500
== Total timerun 3783.0
== PEER INIT
sum 9.0
avg 0.018
== Message count
PRE_INITIATE_MSG 6930
REPLY_PRE_INITIATE_MSG 6930
INITIATE_MSG 418
NOISE_EXCHANGE_MSG 5712
```

```
REPLY_INITIATE_MSG 418
BROADCAST_SUM_MSG 8269
Total messages = 28677
```

Bibliography

[1] Electiricity usage, 2017. URL `http://bwired.nl`. (Accessed: 2017-06-12).

[2] Gergely Ács and Claude Castelluccia. I have a dream!(differentially private smart metering). In *International Workshop on Information Hiding*, pages 118–132. Springer, 2011.

[3] Kostas G Anagnostakis and Michael B Greenwald. Exchange-based incentive mechanisms for peer-to-peer file sharing. In *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, pages 524–533. IEEE, 2004.

[4] Jeffrey A Burke, Deborah Estrin, Mark Hansen, Andrew Parker, Nithya Ramanathan, Sasank Reddy, and Mani B Srivastava. Participatory sensing. *Center for Embedded Network Sensing*, 2006.

[5] Ye Cai, Tao Huang, Ettore Bompard, Yijia Cao, and Yong Li. Self-sustainable community of electricity prosumers in the emerging distribution system. *IEEE Transactions on Smart Grid*, 2016.

[6] Claude Castelluccia, Einar Mykletun, and Gene Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*, pages 109–117. IEEE, 2005.

[7] Ann Cavoukian, Jules Polonetsky, and Christopher Wolf. Smartprivacy for the smart grid: embedding privacy into the design of electricity conservation. *Identity in the Information Society*, 3(2):275–294, 2010.

[8] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing privacy enhancing technologies*, pages 46–66. Springer, 2001.

[9] European Commission. Protection of personal data, 2016. URL `http://ec.europa.eu/justice/data-protection/`. (Accessed: 2017-08-08).

[10] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. *arXiv preprint arXiv:1703.06255*, 2017.

[11] Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015. doi: 10.1017/CBO9781107337756.

[12] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. Efficient and secure comparison for on-line auctions. In *Australasian Conference on Information Security and Privacy*, pages 416–430. Springer, 2007.

[13] Roberto Di Pietro, Stefano Guarino, Nino Vincenzo Verde, and Josep Domingo-Ferrer. Security in wireless ad-hoc networks–a survey. *Computer Communications*, 51:1–20, 2014.

[14] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[15] Roger Dingledine, Michael J Freedman, and David Molnar. The free haven project: Distributed anonymous storage service. In *Designing Privacy Enhancing Technologies*, pages 67–95. Springer, 2001.

[16] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symposium*, 2004.

[17] EU Directive. 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the EC*, 23 (6), 1995.

[18] Zekeriya Erkin. Private data aggregation with groups for smart grids in a dynamic setting using crt. In *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*, pages 1–6. IEEE, 2015.

[19] Zekeriya Erkin and Gene Tsudik. Private computation of spatial and temporal power consumption with smart meters. In *International Conference on Applied Cryptography and Network Security*, pages 561–577. Springer, 2012.

[20] Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. Privacy-preserving face recognition. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 235–253. Springer, 2009.

[21] Laurent Eschenauer and Virgil D Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 41–47. ACM, 2002.

[22] Wojciech Galuba, Karl Aberer, Zoran Despotovic, and Wolfgang Kellerer. Protopeer: a p2p toolkit bridging the gap between simulation and live deployement. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, page 60. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[23] Raghu K Ganti, Fan Ye, and Hui Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11), 2011.

[24] Flavio D Garcia and Bart Jacobs. Privacy-friendly energy-metering via homomorphic encryption. In *International Workshop on Security and Trust Management*, pages 226–238. Springer, 2010.

[25] NIST Smart Grid. Introduction to nistir 7628 guidelines for smart grid cyber security. *Guideline, Sep*, 2010.

[26] George W Hart. Residential energy monitoring and computerized surveillance via utility power flows. *IEEE Technology and Society Magazine*, 8(2):12–16, 1989.

[27] George William Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870–1891, 1992.

[28] Omar Hasan. Paillier java implementation, 2009. URL `http://liris.cnrs.fr/~ohasan/pprs/paillierdemo/Paillier.java`. (Accessed: 2017-09-20).

[29] RB Hiremath, S Shikha, and NH Ravindranath. Decentralized energy planning; modeling and application—a review. *Renewable and Sustainable Energy Reviews*, 11(5):729–752, 2007.

[30] Jaap-Henk Hoepman. Privacy friendly aggregation of smart meter readings, even when meters crash. In *Proceedings of the 2nd Workshop on Cyber-Physical Security and Resilience in Smart Grids*, pages 3–7. ACM, 2017.

[31] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad hoc networks*, 1(2):293–315, 2003.

[32] Deepak Paramashivan Kaundinya, P Balachandra, and NH Ravindranath. Grid-connected versus stand-alone energy systems for decentralized power—a review of literature. *Renewable and Sustainable Energy Reviews*, 13(8):2041–2050, 2009.

[33] Klaus Kursawe. Some ideas on privacy preserving meter aggregation. *Radboud Universiteit Nijmegen, Tech. Rep. ICIS–R11002*, 2011.

[34] Klaus Kursawe, George Danezis, and Markulf Kohlweiss. Privacy-friendly aggregation for the smart-grid. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 175–191. Springer, 2011.

[35] Nicholas D Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T Campbell. A survey of mobile phone sensing. *IEEE Communications magazine*, 48(9), 2010.

[36] Huang Lu, Jie Li, and Mohsen Guizani. Secure and efficient data transmission for cluster-based wireless sensor networks. *IEEE transactions on parallel and distributed systems*, 25(3):750–761, 2014.

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

[38] Evangelos Pournaras, Martijn Warnier, and Frances MT Brazier. A generic and adaptive aggregation service for large-scale decentralized networks. *Complex Adaptive Systems Modeling*, 1(1):19, 2013.

[39] Elias Leake Quinn. Privacy and the new energy infrastructure. *SSRN eLibrary*, 2009.

[40] General Data Protection Regulation. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46. *Official Journal of the European Union (OJ)*, 59:1–88, 2016.

[41] Alfredo Rial and George Danezis. Privacy-preserving smart metering. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, pages 49–60. ACM, 2011.

[42] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 329–350. Springer, 2001.

[43] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[44] Lay Toal. Ordered tree, 2017. URL `http://cs.lmu.edu/~ray/notes/orderedtrees/`. (Accessed: 2017-10-19).

[45] Iman Vakilinia, Jiajun Xin, Ming Li, and Linke Guo. Privacy-preserving data aggregation over incomplete data for crowdsensing. In *Global Communications Conference (GLOBECOM), 2016 IEEE*, pages 1–6. IEEE, 2016.

[46] Alex Vasili. How much electricity could your home generate?, 2017. URL `https://www.theecoexperts.co.uk/how-much-electricity-does-average-solar-panel-system-generate`. (Accessed: 2017-10-09).

[47] Marc Waldman, Aviel D Rubin, and Lorrie Faith Cranor. Publius: A robust, tamper-evident censorship-resistant web publishing system. In *9th USENIX Security Symposium*, pages 59–72, 2000.

[48] Andrew C Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 160–164. IEEE, 1982.

[49] Sencun Zhu, Sanjeev Setia, Sushil Jajodia, and Peng Ning. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In *Security and privacy, 2004. Proceedings. 2004 IEEE symposium on*, pages 259–271. IEEE, 2004.