MINISTRY OF EDUCATION
AND TRAINING

VIETNAM ACADEMY OF SCIENCE
AND TECHNOLOGY

GRADUATE UNIVERSITY OF SCIENCE AND TECHNOLOGY

Vu Duy Hien

DEVELOPING EFFICIENT AND SECURE MULTI-PARTY SUM COMPUTATION PROTOCOLS AND THEIR APPLICATIONS

DISSERTATION ON INFORMATION SYSTEM

Hanoi – 2024

BỘ GIÁO DỤC
VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC
VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Vũ Duy Hiến

NGHIÊN CỨU PHÁT TRIỂN MỘT SỐ GIAO THỨC TÍNH TỔNG BẢO MẬT HIỆU QUẢ TRONG MÔ HÌNH DỮ LIỆU PHÂN TÁN ĐẦY ĐỦ VÀ ỨNG DỤNG

LUẬN ÁN TIẾN SĨ NGÀNH HỆ THỐNG THÔNG TIN

Hà Nội – 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO	VIỆN HÀN LÂM KHOA HỌC VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Vũ Duy Hiến

NGHIÊN CỨU PHÁT TRIỂN MỘT SỐ GIAO THỨC TÍNH TỔNG BẢO MẬT HIỆU QUẢ TRONG MÔ HÌNH DỮ LIỆU PHÂN TÁN ĐẦY ĐỦ VÀ ỨNG DỤNG

LUẬN ÁN TIẾN SĨ NGÀNH HỆ THỐNG THÔNG TIN
Mã số: 9 48 01 04

Xác nhận của Học viện Khoa học và Công nghệ	**Người hướng dẫn 1** *(Ký, ghi rõ họ tên)*	**Người hướng dẫn 2** *(Ký, ghi rõ họ tên)*
	GS. TSKH. Hồ Tú Bảo	**PGS. TS. Lương Thế Dũng**

Hà Nội - 2024

PLEDGE

I promise that the thesis: ”*Developing efficient and secure multi-party sum computation protocols and their applications*” is my original research work under the guidance of the academic supervisors. All contents of the thesis were written based on papers and articles published in distinguished international conferences and journals published by the reputed publishers. The source of the references in this thesis are explitly cited. My research results were published jointly with other authors and were agreed upon by the co-authors when included in the thesis. New results and discussions presented in the thesis are perfectly honest and they have not yet published by any other authors beyond my publications. This thesis has been finished during the time I work as a PhD student at Graduate University of Science and Technology, Vietnam Academy of Science and Technology.

Hanoi, 2024

PhD student

Vu Duy Hien

ACKNOWLEDGEMENTS

Scientific research is an interesting journey where the thesis is one of the first results that researchers have reached. On that journey, I have met many kind people who have supported for me to finish this thesis.

First of all, I would like to thank my great supervisors **Prof. Dr. Ho Tu Bao** and **Assoc. Prof. Dr. Luong The Dung** who have provided valuable advice to me. Without their support and guidance, I would not able to complete my thesis. I have learned a lot of things from my supervisors.

I am thankful to **Graduate University of Science and Technology, colleagues at Banking Academy of Vietnam, friends**, and **collaborators** who always encourage me along my research journey.

I also thank **the CAMEL cafe (No.**104/1 **Viet Hung street, Long Bien district, Ha Noi)** where my publications and thesis had been born in.

Finally, I want to send the most special thank to **my big family, my wife**, and **our children** who always have my back.

Hanoi, 2024

PhD student

Vu Duy Hien

CONTENTS

LIST OF ABBREVIATIONS

BoW	Bag-of-Words
CDH	Computational Diffie-Hellman
DDH	Decisional Diffie-Hellman
DD-PKE	Public-key encryption with a double-decryption algorithm
DNA	Deoxyribonucleic acid
DRE	Direct-recording electronic
DSS	Digital signature standard
E2E	End-to-end
LWE	Learn with error
NSC	National university of Singapore short text messages corpus
PPFC	Privacy-preserving frequency computation
PPML	Privacy-preserving machine learning
PPNBC	Privacy-preserving Naive Bayes classification
PSI	Private set intersection
RAM	Random Access Machines
SMC	Secure multi-party computation
SMS	Secure multi-party sum
SSC	Secure sum computation
TF-IDF	Term frequency – inverse document frequency
UK	United Kingdom
ZKP	Zero knowledge proof

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

A. Motivation

Nowadays, the development of information technology and communication, especially the birth of web applications or information systems has created a large amount of data owned by organizations or individuals. This has spurred the development of the distributed computing field where the data owners perform together computational tasks based on their cooperative data [1, 2]. Basically, the distributed computing field has brought a lot of substantial benefits to organizations and individuals, such asreducing significantly costs, understanding comprehensively customers, and making good business decisions. However, in fact, because of privacy policy or business secrets, participants of distributed computing systems often wish to obtain cooperative tasks' correct output without revealing their input data. For instance, some banks cooperate together to improve machine learning-based credit scoring tool using their customers' data, but they are not ready to share their customers' data for anyone. Similarly, although there are some hospitals who want to jointly develop disease diagnosis methods based on a large united database, however they do not want to provide their patients' data to others. These challenges had motivated the birth of `SECURE MULTI-PARTY COMPUTATION` area (`SMC`, for short) that has been considered as a subfield of modern cryptography.

In essence, `Secure Multi-party Computation` refers distributed computing methods in security concerns [1, 3]. Particularly, in a secure multi-party computation model, there are several parties, in which each participant owns a private input. These participants wish to obtain the result of the specific function f over all private inputs while each party reveals nothing about his/her input but the output result. Unlike traditional cryptography field, the adversary of `SMC` problems in general and the `SMS` problem in particular can be inside the system of participants. The attacks of the adversary may be to learn the honest participants' private input or to cause the outputs to be incorrect [1]. As a result, the "`secure`" term here means: *(1) the output's cor-*

rectness is guaranteed, and *(2) each party's input is privately kept by himself/herself.*

Nowadays, SMC has become an interesting topic that has attracted more and more attention from research community. A variety of SMC problems have been formulated and their solutions have been proposed into SMC protocols, such as secure comparison protocols [4,5], secure multi-party sum computation protocols [6–8], and secure dot product protocols [6,9–11]. Furthermore, such SMC protocols have been applied to various practical problems, such as secure online auction [14], secure e-voting systems [12,13], privacy-preserving queries system [15], privacy-preserving financial data analytic [16], privacy-preserving online advertising [17], and privacy-preserving machine learning/data mining [18–20].

This thesis has investigated one of the most important and popular SMC problems [6] that is the secure multi-party sum computation one (SMS, for short). In the SMS problem, it is assumed that where there are some parties, in which each party owns a private value as his/her input, and the parties wish to obtain the sum of all inputs but they reveal nothing about their inputs beyond the sum value. Similarly to SMC problems in general, the birth of SMS one has been based on the security requirements of specific distributed computing problems. Currently, a lot of protocols have been propounded for the SMS problem, and they have a wide applicability in various practical computing tasks, such as privacy-preserving recommendation system [21], privacy-preserving multi-party data analytics [22], secure electronic voting system [12, 13], privacy-preserving association rule mining [6, 7], privacy-preserving classification [23], secure data collection for the smart grid [24], and secure auction [25, 26].

For SMC problems in general, and SMS one in particular, the protocols must be secure (mainly including the preservation of the privacy of the participants' local inputs and the correctness of the honest parties' outputs [3]) enough to prevent the adversary's harmful behaviors. Besides, SMS protocols should be good performance (i.e. low computational complexity and communication cost) to be implemented in real-life applications. This is perfectly understandable, because a lot of practical SMS problems require to perform computational tasks as quickly as possible, such as secure e-voting, secure online auction. `SMS` protocols-based privacy-preservation solutions such as privacy-preserving Apriori algorithm for mining association rules, privacy-preserving Naive Bayes classifier, and secure gradient descent algorithm have to execute `SMS` protocol multiple times to compute necessary mediate values. Moreover, in many distributed computing scenarios, participants use devices limited in computational ability, storage capacity, and connectivity, e.g. smartphones, tablets. Thus, it is significant to develop `SMS` protocols having both high security level and good performance.

B. Research objectives

As mentioned before, first of all, `SMS` protocols need to be secure. To do this, `SMS` protocols either (1) require each participant to split his/her private value into a number of parts, and he then shares them with all others using secure communication channels or (2) use homomorphic cryptosystems such as ElGamal encryption scheme [27] or Paillier cryptosystem [28]. Considering the approach (1), such protocols obviously have high cost of communication, and they are unsuitable for multi-party computational models with a large number of participants. In contrast, `SMS` protocols based on the second approach (2) often have pricey cost of computation. As a result, it can be stated that the biggest challenge for designing `SMS` protocols is how to create `SMS` protocols having both high security level and good performance. Thus, the research objectives of this thesis include:

- Designing efficient and secure multi-party sum computation protocols that have the capability to preserve the privacy of the parties' local inputs and the correctness of the honest parties' outputs, as well as good performance.
- Developing `SMS`-based solutions for practical problems that have been currently solved by existing `SMS` protocols but are not yet secure and efficient.

C. Main contributions

The scientific story of this thesis is narrated as follows:

- The thesis starts with basic distributed computing problems requiring to execute `SMS` protocols once (e.g. the single-candidate secure e-voting problem). Through a comprehensively analysis, one of the most typical `SMS` protocols has been chosen to be re-designed. The improved `SMS` protocol is then optimized by transforming into the elliptic curve analog of the ElGamal cryptosystem-based variant. Hence, the first proposed protocol has not only high level of security, but also good performance. Continuously, based on one of the most typical `SMS` protocols mentioned above, the thesis tries to integrate a Schnorr signature-derived authentication method into a secure multi-party sum computation function, in which both these cryptographic tools employ the same private and public keys. Hence, the second proposed protocol has a unique feature which is unlike the existing work, that is no need to pre-establish any authenticated channel between each tuple of parties. Furthermore, this protocol is still secure in the common semi-honest model, as well as efficient in real-life applications.

- In the next stage, the thesis considers practical problems where `SMS` protocols have been performed multiple times for solving specific distributed computing tasks (e.g. privacy-preserving data mining and machine learning problems). The selected typical `SMS` protocol is re-designed with the aim of obtaining many sum values only in one round of computation and communication. As a result, the third proposed protocol efficiently computes multiple sum values. In addition, this proposal significantly saves the cost of key generation and management.

- Finally, to demonstrate the applicability of the above results, the thesis constructs the new protocols-based solutions for the secure end-to-end e-voting scheme and the privacy-preserving Naive Bayes classification problem in the horizontal dataset setting.

The general contribution of this thesis is to propose novel `SMS` protocols. However, unlike the previous work, the `SMS` protocols of this thesis are efficient to be implemented in real-life applications.

In particular, the contributions of this thesis are presented in the following sections.

The first contribution

The thesis proposes three novel SMS protocols based on the homomorphic ElGamal encryption. Because this standard cryptography technique is semantically secure, all proposed protocols achieve a high level of security without using any trusted party or more than two non-colluding parties. Three new SMS protocols include:

- The privacy-preserving frequency computation (PPFC) protocol that can obtain a frequency value in the context where communication channels among parties are authenticated. In addition to high level of security, this protocol has good performance, since it is optimally re-designed from the ideas of the typical SMS ones and the elliptic curve cryptography. Consequently, the proposed PPFC protocol can be employed as a key building block to securely and rapidly compute single or multiple sum values (e.g. counting the result of secure e-voting problems).

- The SMS protocol can securely compute a sum value in the scenario where communication channels among parties are only public. This proposal is methodically combined of a secure sum function and a Schnorr signature-derived authentication method, so the second SMS protocol not only satisfies the mandatory requirement of security, but also is efficient. Especially, this protocol can be directly implemented on public channels (e.g. Internet) without pre-establishing any authenticated/secure channels. Because of the above advantages, the second SMS protocol can become a suitable solution for the secure single-candidate electronic voting problem in the semi-honest model.

- The secure multi-sum computation protocol that can privately compute multiple sum values in one round of computation and communication. By using an optimal technique for solving discrete logarithm problems with small space of solutions, this protocol has not only a high security level but also good performance.

The second contribution

Based on analysis of the proposed protocols' applicability and essential requirements of practical problems, the second contribution is to develop secure and efficient solutions for the secure electronic end-to-end voting scheme and the privacy-preserving Naive Bayes classifier in the horizontally distributed scenario. Particularly, because the secure electronic end-to-end voting scheme often require to accurately and rapidly count the voting result over various types of communication channel, the combination of the proposed PPFC and the `SMS` protocols are chosen to solve this problem. For the privacy-preserving Naive Bayes classifier that requires to sum up frequency values used for constructing the Naive Bayes classification model while the parites reveal nothing about their data, the thesis employs the secure multi-sum computation protocol for boosting this highly complex task.

D. Thesis organization

The main content of this thesis is organized as follows:

- Chapter 1 provides a general background about secure multi-party computation such as basic concepts, definition of security, and cryptography preliminaries. After that, this chapter of the thesis comprehensively reviews related work to identify research gap and new directions.
- Chapter 2 analyzes typical `SMS` protocols in detail. Based on the analysis result, this chapter proposes three new protocols for privacy-preserving frequency computation, secure multi-party sum computation without pre-establishing secure/authenticated channels, and secure multi-sum computation problems.
- Chapter 3 develops the solutions based on the new `SMS` protocols for two practical applications, i.e. the secure electronic voting scheme and the privacy-preserving Naive Bayes classifier.

CHAPTER 1. OVERVIEW OF SECURE MULTI-PARTY SUM COMPUTATION

In this chapter, the thesis first provides a background of secure multi-party computation field. Next, this chapter introduces the secure multi-party sum computation problem, then the previous work closely related to this problem is meticulously analyzed that supports to identify potential research issues.

1.1. Background of secure multi-party computation

1.1.1. Introduction

As mentioned before, `Secure Multi-party Computation` (as illustrated in Figure 1.1) refers distributed computing methods in security concerns [1,3], in which:

- *Input*: there are n parties where each participant i owns a private input v_i.
- *Output*: the participants obtain the result $f(v_1,...,v_n)$ of the specific function f over the inputs $(v_1,...,v_n)$, and each party reveals nothing about his/her input but the output result.

Here, it needs to be expressed that the "`secure`" concept means the two following constraints:

- The correctness of the function's output is guaranteed.
- Each party's input is privately kept by himself/herself.

Generally, the security property of a `SMC` protocol depends on the adversary model including type of adversary (i.e. semi-honest or malicious), type of communication channels (i.e. secure, authenticated, or public), and capabilities of adversary (i.e. number of controlled parties, eavesdropping transferred messages, and computational power). Hence, the design of a `SMC` protocol needs to achieve the security level corresponding to the selected adversary model. This aspect is fully analyzed in the next sections.

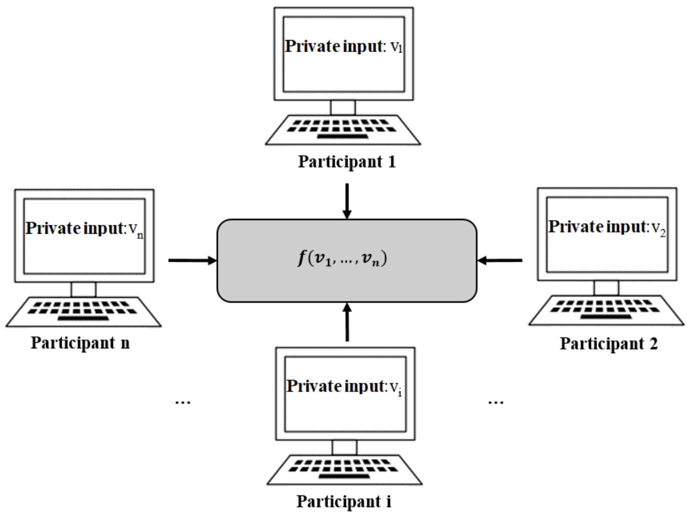


Figure 1.1: The distributed computing model in a secure manner

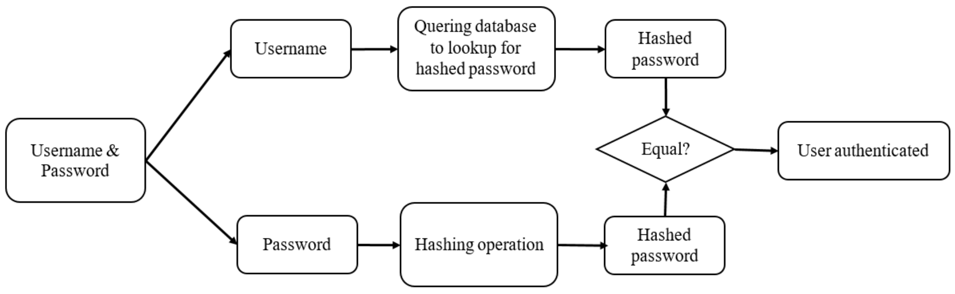


Figure 1.2: An example of the authentication method without knowing user's password

It can be seen that there are many practical problems related to `SMC`. A highly popular `SMC` problem is the authentication method as illustrated in Figure 1.2 where a server has to obtain the output of user verification function (i.e. "true/false") while this server does not store the user's passwords into database (of course, they cannot know what the user's passwords are).

Also related to the issue of password management, `Apple Inc.` [29] monitors the user's passwords by securely matching such passwords (privately stored in the autofill keychain on the user's local device) against a large set of weak or leaked passwords. As depicted in Figure 1.3, `Apple`'s technologies can detect the user's passwords occurring on the list of weak or leaked passwords (e.g. `12345678`, `password`, and `iloveyou`) without knowing what the user's passwords are.

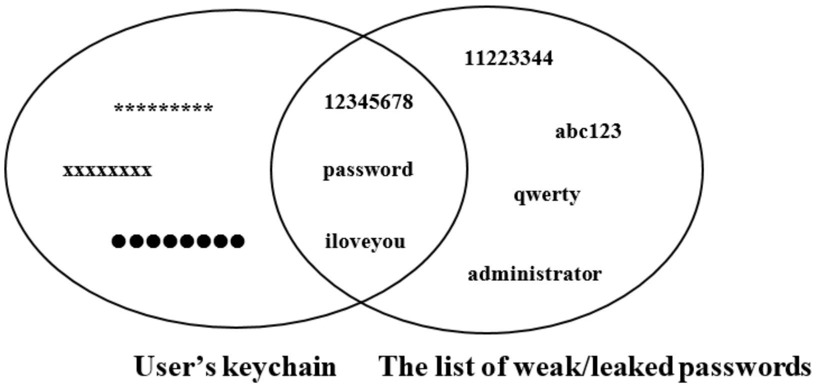


Figure 1.3: An example of monitoring user's passwords

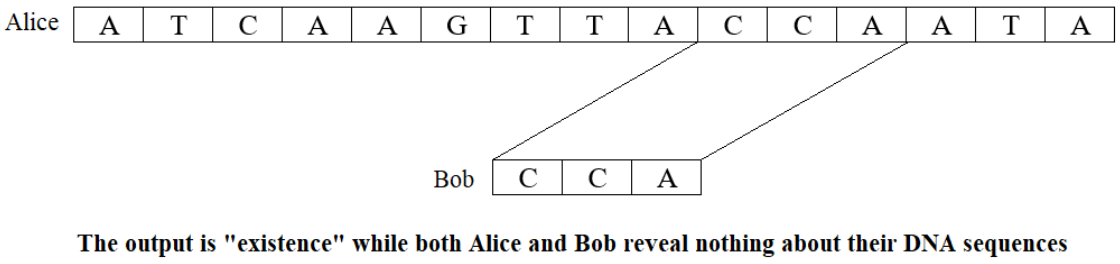

Alice
A T C A A G T T A C C A A T A
Bob
C C A
The output is "existence" while both Alice and Bob reveal nothing about their DNA sequences

Figure 1.4: An example of the DNA pattern-matching problem

Considering the DNA pattern-matching problem [30] (as illustrated in Figure 1.4), there are a party who wants to determine a specific DNA subsequence's existence (e.g. a short DNA string that describes a mutation leading to a disease) inside a DNA sequence owned by another party without disclosing to each party's input.

Another typical SMC problem as depicted in Figure 1.5 is the sealed-bid auction system where the auctioneer exactly determines the winner without opening the bids.

In general, the solutions for SMCproblems have been formulated into SMCprotocols that have been defined as a set of specific rules and guidelines for processing, computing, and communicating data among participants.

Nowadays, SMC has become an interesting topic that has attracted more and more attention from research community. Hence, a lot of protocols have been proposed for different SMC problems, such as secure comparison protocols [4, 5], secure multi-party sum computation protocols [6–8], and secure dot product protocols [6, 9–11]. Furthermore, such SMC protocols have been applied to various practical prob-

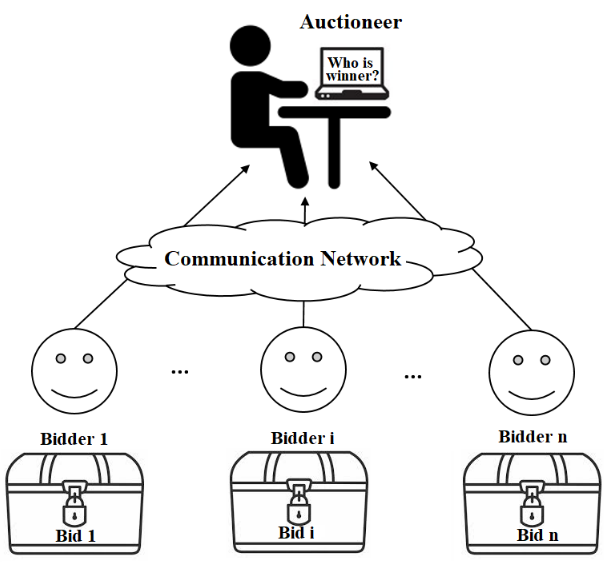


Figure 1.5: The secure electronic sealed-bid auction model

lems, such as secure e-voting systems [12, 13], secure online auction [14], privacy-preserving queries system [15], privacy-preserving financial data analytic [16], privacy-preserving online advertising [17], and privacy-preserving machine learning and data mining [18–20].

1.1.2. Basic concept

A general SMC problem is formulated as follows [3].

Let n ($n \geq 2$) be the number of participants joining a distributed computing network, in which the i^{th} party keeps a private input v_i ($i = 1, ..., n$), and all inputs have the same length ($|v_i| = |v_j|$ with $\forall i, j$). The multi-party computation function f is defined as follows:

$$\boxed{\begin{array}{rcl} f : (\{0,1\}^*)^n & \rightarrow & (\{0,1\}^*)^n \\ \bar{v} = (v_1, ..., v_n) & \rightarrow & f(\bar{v}) = (f_1(\bar{v}), ..., f_n(\bar{v})) \end{array}} \tag{1.1.1}$$

As depicted above in Figure 1.1, the i^{th} party who owns the private input value v_i wishes to obtain the i^{th} element in $f(v_1, ..., v_n)$ that is $f_i(v_1, ..., v_n)$ (denoted as y_i).

A multi-party computation function f can fall into one of the following types:

- *Deterministic functions*: that return a unique output with the same input value, and include:
    - *Symmetric deterministic functions*: that are deterministic functions in which $f_i(v_1,...,v_n) \equiv f_j(v_1,...,v_n)$ with $\forall i \neq j$.
    - *Asymmetric deterministic functions*: that are deterministic functions where $f_i(v_1,...,v_n) \neq f_j(v_1,...,v_n)$ with $\forall i \neq j$.
- *General functions (including both deterministic and indeterministic functions)*: that can return different outputs with the same input value in different executions.

Conceptually, the secure multi-party computation field refers to methods that allow the participants to securely compute a function f based on their private inputs while anyone learns nothing about each party's input.

In essence, the SMC area is perfectly close to the traditional cryptography field, because the design of a basic cryptographic scheme (e.g. encryption, digital signature) in a multi-party environment can be viewed as the design of a SMC protocol for solving a specific issue [3], i.e. confidentiality, authentication, or integrity [2, 3]. Thus, the SMC area has become a crucial part of the modern cryptography [3]. In the opposite perspective, there still exists the difference between the traditional cryptography field and the SMC area [3]. This is explained that the basic cryptographic primitives (e.g. encryption, digital signature) require participants to perform little interaction while SMC protocols' parties are often have to interact with others multiple times.

Next, the thesis provides a well-known security definition of a general secure multi-party computation protocol.

1.1.3. Definition of security

Before representing the standard definition of security for the SMC field, the thesis describes an adversary model chosen for this study, a general approach formalizing the security of a SMC protocol, and necessary technical preliminaries.

1.1.3.1. Adversary model

This section formalizes possible attacks on a SMC protocol into an adversary model that has been used as an important basis to design provable secure cryptographic protocols. Referred from the work [31], the adversary model of this study also consists of three components, i.e. assumptions, goals, and capabilities of an adversary.

i. Adversary assumptions

Basically, one of the most different characterizes between the SMC field with the traditional cryptography (e.g. encryption, digital signature) that a SMC protocol can be attacked by not only an external entity but also a set of the corrupted internal parties controlled by an external entity [3]. Consequently, the computational model of SMC includes three types of entity: *(1) honest parties* who follow the rule of protocol and they do not collude with any one to perform malicious behaviors, *(2) corrupted parties* who are ready to collude with others or can be controlled by an external adversarial entity to execute malicious behaviors against honest parties, and *(3) external adversary* who controls corrupted parties to perform malicious behaviors.

Considering the corrupted parties' behaviors, if the corrupted parties are semi-honest, then they still follow the protocol's rule but they can collude together or be controlled by the adversary to execute the harmful behaviors such as trying to gain others' private data input. In contrast, in the case the corrupted parties are malicious, they can arbitrarily perform their behaviors without following the protocol's rule, even may abort the protocol anytime. Based on the corrupted parties' behaviors, there are two types of SMC model, i.e. the *semi-honest* and *malicious* models.

This thesis focuses on the semi-honest model, and the number of corrupted parties is up to $(n-2)$ where n is the number of data users participating the protocol execution. SMC protocols based on the semi-honest model are quite efficient, so this model is suitable for applications requiring high performance, such as privacy-preserving distributed data mining and analytic [32–34]. It can be understandable, because if a party who is ready to participate in a SMC protocol execution with his goodwill and reputation, then he should follow the rule of protocol. For example, there are several hospitals who wish to jointly research on their united patient records. Due to privacy constraints, each hospital is not allowed to know others' data. Clearly, the semi-honest model is appropriate for such scenario. In the case there exist curious parties who want to discover others' private data based on what they observed, they should be prevented by the protocol's design.

Here, it should be emphasized that the parties in the semi-honest model only adhere to the rules of computation, so that the non-collusion assumption (e.g. in [23, 35–37]) is unreasonable [1]. It is also noted that although the security requirement of SMC protocols in the semi-honest model is not too strict, this model is an important first step toward achieving higher levels of security. The semi-honest model thus will play a major role in the design of protocols for the malicious model, and it can be transformed protocols that are secure in the semi-honest model into protocols that are secure in the malicious model [3].

Next, because of controlling the corrupted internal parties, it is assumed that the adversary knows the corrupted internal parties' knowledge (e.g. private keys, confidential data input), as well as accessing communication channels among parties. In addition, to consolidate the contributions, this thesis assumes that the communication channels between the parties are only authenticated or even public.

ii. Adversary's goals

While the classical distributed computing field often face inadvertent threats such as unstable communication and machine crashes, SMC protocols are concerned with some adversarial entity's attacks with the aims of learning the honest parties' private input or causing the output result to be incorrect [1].

iii. Adversary's capabilities

As mentioned before, the adversary has an extremely powerful capability that controls up to $(n-2)$ corrupted internal parties (of course, we cannot know who the honest parties are) to perform malicious behaviors. Because the communication channels between parties are authenticated or even public, the adversary can eavesdrop transferred messages. Besides, it is assumed that the adversary is computationally bounded, that is, it runs in (probabilistic) polynomial-time [1].

1.1.3.2. Definitional approach

The direct way to define the security of a SMC protocol is to predetermine the requirements, then show that the protocol satisfies all of them [3, 38]. However, this approach is not general because: (i) an important property can be ignored, (ii) the security definition is simple enough to see that the adversary's possible attacks can be prevented [1].

To choose a suitable approach for defining security of SMC protocols, let us begin with a very basic paradigm for a public-key cryptosystem, that is semantic security. Goldwasser and Micali [39] stated that a public-key cryptosystem is semantically secure if whatever an adversary can compute about the plaintext given the ciphertext, then it can also compute when receiving nothing. Obviously, if the adversary receives nothing, then it gains nothing about the plaintext. The context where the adversary receives nothing seems to imply an ”ideal world” [40]. Explicitly speaking, a system is secure in the real world, if the adversary receives the ciphertext but nothing is learned (equivalent to the ideal world where the adversary receives nothing). More generally, the security of a system is proved by comparing what happens in the ”real world” to what happens in the ”ideal world”. As a result, this formulation of security is called the ”ideal/real simulation paradigm”. Moreover, the simulation-based security model is the simplest but the most rigorous among the security models for malicious adversaries.

For the secure multi-party computation field (see Figure 1.6), the ideal world model is where there exists a trusted party who helps the participants to compute the output without security concerns, and the real one is where no trusted party exists. In the other words, every participant does not trust anyone in the real world. The security of a protocol is determined by comparing the outcome of a real protocol execution to

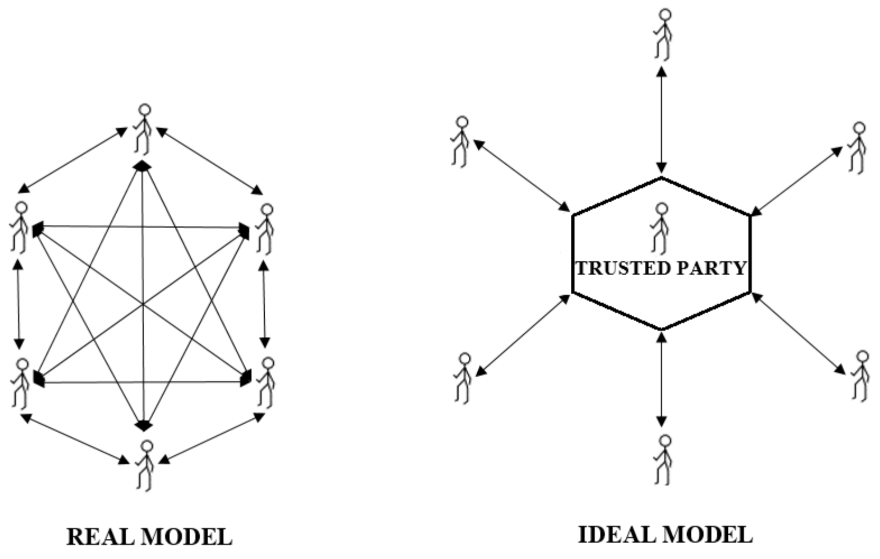


Figure 1.6: The real and ideal models in distributed computing field

the one of an ideal protocol execution [3].

Thus, in SMC field, the simulation-based security model has been used as an important approach for proving a SMC protocol's security.

1.1.3.3. Technical preliminaries

This section represents some necessary preliminaries employed for the SMC field's standard security definition.

i. Negligible function

Let n be a security parameter (well-known as the key length which the hard problems such as discrete logarithm, large integer factorization cannot be solved in poly-nominal time). Below is the definition of a negligible function referred from the book [3].

Definition 1.1.1. *A function $\mu(u)$ is called negligible with n if for all positive polynomial $p(.)$, there exists a non-negative integer N such that $\forall n > N$:*

$$\mu(u) < \frac{1}{p(n)} \tag{1.1.2}$$

ii. Computationally indistinguishable

The notion of computational indistinguishability is very crucial for both the cryptography and SMC field [41]. Hence, the following definition [3] is provided.

Definition 1.1.2. *Let $X(n,a), Y(n,a)$ be two random ensembles indexed by (n,a) and $X = \{X(n,a)\}_{n\in\mathbb{N}, a\in\{0,1\}^*}, Y = \{Y(n,a)\}_{n\in\mathbb{N}, a\in\{0,1\}^*}$ are corresponding distributions. X, Y are called "computationally indistinguishable" (denoted as $X \overset{C}{\equiv} Y$) in polynominal time if every probabilistic polynomial-time algorithm D, there exists a negligible function $\mu(u)$ with n such that $\forall a \in \{0,1\}^*$:*

$$|Pr[D(X(n,a)) = 1] - Pr[D(Y(n,a)) = 1| < \mu(u) \qquad (1.1.3)$$

In SMC field, the above parameters can be understood as follows:

- n is security parameter.
- a is the input of SMC protocols.
- X is the output of SMC protocols in ideal world setting.
- Y is the output of SMC protocols in real world setting.

1.1.3.4. Standard definition of security

According to the simulation-based approach, this section presents the standard definition of security of a SMC protocol in the semi-honest model using public channels that is referred from the SMC framework [3].

Definition 1.1.3. *(privacy with respect to the semi-honest model using public channels [3])*

Let f be a secure multi-party computation function as defined in Section 1.1.1.

- *In the case f is a deterministic function: the protocol Π privately computes the function f against t corrupted participants if $\forall I \subseteq \{1,2,...,n\}$ such that $||I|| = t$, there exists a probabilistic polynomial-time algorithm $\mathbb{M}$ such that*

$$\boxed{\{\mathbb{M}(I,\overline{v}_I,f_I(\overline{v}))\}_{\overline{v}\in(\{0,1\}^*)^n} \overset{c}{\equiv} \{VIEW^{\Pi}_{A,I}(\overline{v})\}_{\overline{v}\in(\{0,1\}^*)^n}} \tag{1.1.4}$$

• *In general case: the protocol* Π *privately computes the function* f *against* t *corrupted participants if* $\forall I \subseteq \{1,2,...,n\}$ *such that* $\|I\| = t$*, there exists a probabilistic polynomial-time algorithm* $\mathbb{M}$ *such that*

$$\boxed{\{\mathbb{M}(I,\overline{v}_I,f_I(\overline{v}),f(\overline{v}))\}_{\overline{v}\in(\{0,1\}^*)^n} \overset{c}{\equiv} \{VIEW^{\Pi}_{A,I}(\overline{v}),OUTPUT^{\Pi}(\overline{v})\}_{\overline{v}\in(\{0,1\}^*)^n}} \tag{1.1.5}$$

where

• $VIEW^{\Pi}_{A,I}(\overline{v})$ *is the views of* t *corrupted participants and all messages (transferred among the honest participants) that the adversary* A *eavesdrops during the execution protocol* Π *on the input* $\overline{v} = (v_1,...,v_n)$

• $OUTPUT^{\Pi}(\overline{v})$ *is the output sequence of all parties involving the protocol* Π*. In the first case,* $OUTPUT^{\Pi}(\overline{v}) \equiv f(\overline{v})$

• $\overset{c}{\equiv}$ *is computational indistinguishability.*

Besides, there is a composition theorem often used to construct SMC protocols in the semi-honest model (see Theorem 1.1.1).

Theorem 1.1.1. *Suppose that the function* g *is privately reducible to the function* f*, and* f *is privately computed by a secure protocol. Then there exists a protocol for privately computing* g *[3].*

This theorem says that if a protocol can be decomposed into sub-protocols, then it will be secure if its sub-protocols are secure [3].

In this thesis, all proposals' security is proved using Definition 1.1.3 and Theorem 1.1.1.

Next, the thesis presents foundation of cryptography used as preliminaries of secure multi-party computation field.

1.1.4. Cryptographic preliminaries

1.1.4.1. Discrete logarithm problems

For general cryptographic protocols, the discrete logarithm problems can be seen as one of the most important preliminaries. As a result, this section provides basic concepts related to the discrete logarithm problems referred from the book [41].

Considering a cyclic group $\mathbb{G}$ of order q ($\mathbb{G} = \{g^0, g^1, \ldots, g^{q-1}\}$). This equals to $\forall h \in \mathbb{G}$, there only exists a unique value $x \in \mathbb{Z}_q$ such that $g^x = h$. In that context, it can be called "x is discrete logarithm of h with the base g" and written $x = log_g h$. The hard discrete logarithm problem is defined as follows:

Definition 1.1.4. *[41] Let $\mathbb{G}$ be a cyclic group of order q ($\|q\| = n$) with the generator g and a random element $h \in \mathbb{G}$. The discrete logarithm problem in $\mathbb{G}$ is to compute $log_g h$. The experiment simulating the discrete logarithm problem in $\mathbb{G}$ (denoted as $DLog_{A,G}(n)$) is described in the following steps:*

- *Run the poly-nominal algorithm $G(1^n)$ to obtain the parameters $(\mathbb{G}, q, g)$.*
- *Choose a random element $h \in \mathbb{G}$.*
- *The algorithm A is given $(\mathbb{G}, q, g, h)$ and output the value $x \in \mathbb{Z}_q$.*
- *If $g^x = h$, then the output of this experiment is 1. And 0 if otherwise.*

The discrete logarithm problem is hard relative to $\mathbb{G}$, if for all probabilistic polynomial-time algorithms A, then there exists a negligible function $\mu(n)$ such that

$$Pr[DLog_{A,G}(n)] < \mu(n) \tag{1.1.6}$$

Informally, although the algorithm A is given $(\mathbb{G}, q, g, h)$, the probability for A to find out $x \in \mathbb{Z}_q$ satisfying $g^x = h$ is negligible.

The problems related to compute discrete logarithms consist of the computational Diffie-Hellman (CDH) and the decisional Diffie-Hellman (DDH) ones.

- ***Computational Diffie-Hellman (CDH) problem***

Given the parameters $(\mathbb{G}, q, g)$ and two elements $h_1 = g^{x_1}, h_2 = g^{x_2}$ belongs to $\mathbb{G}$. $DH_g(h_1, h_2)$ is defined as $\overset{def}{=} g^{x_1 x_2}$. The CDH problem is to compute $DH_g(h_1, h_2)$ given h_1, h_2. If the discrete logarithm problem in $\mathbb{G}$ is easy, then the CDH problem is solved. However, if the CDH problem is hard, then it cannot be stated that the discrete logarithm problem is too. Thus the CDH assumption has seldom used in the cryptography field.

- ***Decisional Diffie-Hellman (DDH) problem***

Given the parameters $(\mathbb{G}, q, g)$ and three elements $X = g^x, Y = g^y, Z = g^z$ with x, y, z are randomly chosen in $\mathbb{Z}_q$. The hard decisional Diffie-Hellman problem is defined as follows:

Definition 1.1.5. *The DDH problem is hard relative to $\mathbb{G}$ if for all probabilistic polynomial-time algorithms A, then there exists a negligible function $\mu(n)$ such that*

$$|Pr[A(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - Pr[A(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]| < \mu(n) \qquad (1.1.7)$$

Basically, this definition states that (g^x, g^y, g^z) and (g^x, g^y, g^{xy}) are computationally indistinguishable with x, y, z are randomly chosen in $\mathbb{Z}_q$. Therefore, the hard DDH problem is a strong assumption commonly used in the cryptography field.

In the SMC field, the computations of discrete logarithm-based protocols are usually performed in cyclic groups of large prime order, because of the following reasons:

- The discrete logarithm problem is hardest in these groups, and the decisional Diffie-Hellman assumption is also held in such groups.
- It is easy to choose a generator of a cyclic group of large prime order (i.e. every element, excepting the identity).

Additionally, cyclic groups of large prime order are suitable for SMC models with large number of parties. In such scenarios, the public parameters $(\mathbb{G}, q, g)$ only need to be generated once, each participant can privately choose his/her confidential parameters.

As a result, the cryptographic parameters for discrete logarithm-based protocols are chosen as follows:

- Let p and q be two large primes such that $(p-1) \vdots q$, and $g \in \mathbb{Z}_p$ satisfying $g \neq 1$ and $g^q \mod p = 1$.
- $\mathbb{G} = \{g^0, g^1, ..., g^{q-1}\}$.
- The public parameters are $(\mathbb{G}, p, q, g)$.

1.1.4.2. ElGamal public-key cryptosystem: a homomorphic encryption

This section represents a common variant of the ElGamal encryption scheme [27] that is based on discrete logarithm problems.

Let $\mathbb{G}, q, g$ be secure cryptographic parameters. In addition, let x be a private key, and the public key is $h = g^x$.

In the encryption step, the sender uses h to create the ciphertext C from the plaintext m by randomly choosing k from $\{1, 2, ..., q-1\}$ and computing the ciphertext $C = (C_1 = mh^k, C_2 = g^k)$. To find out the plaintext m from the ciphertext C, the receiver uses the private key x and computes $m = C_1(C_2^x)^{-1}$.

Under necessary assumptions, the ElGamal encryption is semantically secure. Hence, this cryptosystem has been used to construct several secure cryptographic protocols such as the ElGamal digital signature [27], the Schnorr signature scheme [42]. Moreover, the ElGamal encryption has homomorphic property that is the most important property used for designing SMC protocols.

- *Multiplicative homomorphic property*: it can be seen that if $C(m_1) = (m_1 h^{k_1}, g^{k_1})$ and $C(m_2) = (m_2 h^{k_2}, g^{k_2})$ are the corresponding ciphertexts of m_1, m_2, then $C(m_1)C(m_2) = (m_1 m_2 h^{k_1+k_2}, g^{k_1+k_2})$ is the ciphertext of $m_1 m_2$.
- *Additive homomorphic property*: in the cases the size of plaintexts is not

too large (e.g. input values of SMC problems), the ciphertexts of m_1, m_2 can be modified into $C(m_1) = (g^{m_1}h^{k_1}, g^{k_1}); C(m_2) = (g^{m_2}h^{k_2}, g^{k_2})$, respectively. Consequently, the value $C(m_1)C(m_2) = (g^{m_1+m_2}h^{k_1+k_2}, g^{k_1+k_2})$ is the ciphertext of $(m_1 + m_2)$. Simultaneously, the small-sized value m can be easily extracted from g^m without spending much time, because there exist a lot of methods solving this problem, in which the Shanks' baby-step giant-step algorithm is one of best candidates.

In addition, there exists an elliptic curve analog of the ElGamal cryptosystem [43] described as follow:

Let $q, E(F_q), O, G, q$ be secure cryptographic parameters. The private key is $d \in [1, q-1]$, and the public key $Q = dG$.

To encrypt m, the sender employs the public key Q to compute the ciphertext C by randomly choosing k from $[1, q-1]$ and computing $C = (C_1 = P_m + kQ, C_2 = kG)$ where P_m is a point of E corresponding to the plaintext m (using a method of imbedding plaintexts mentioned in [43]). To decrypt the ciphertext C based on the private key d, the receiver needs to compute the value m decoded from the point M (using a method of imbedding plaintexts mentioned in [43]), in which $M = C_1 + (-dC_2)$.

Under necessary assumptions, the elliptic curve analog of the ElGamal cryptosystem is also semantically secure.

It can be seen that in secure multi-party computing models using the ElGamal encryption, the cryptography parameters ($\mathbb{G}$, g, q) or ($E(F_q)$, O, q, G) can be publicly chosen based on the highest standard of security without using any trusted third party, and each participant only needs to choose private keys for himself/herself. Hence, the ElGamal cryptosystem is suitable for such multi-party computing models. Because of the advantages above, the ElGamal encryption is regarded as a key building block of this thesis.

1.1.4.3. Solving discrete logarithm problems with small space of solutions

For the ElGamal cryptosystem-based SMC protocols, we often face discrete logarithm problems in which their solution space is limited by small or medium values. Basically, there are two common methods to solve discrete logarithm problems: Brute-force and Shanks' baby-step giant-step algorithms (see Appendices A and B).

1.2. Secure multi-party sum computation problem

1.2.1. Problem formulation

As illustrated in Figure 1.7, the SMS problem is formulated as follows:

- *Input*: there are n parties, in which each participant i owns a private value v_i.
- *Output*: the participants obtain the sum $f(v_1,...,v_n) = v_1 + ... + v_n$, and each party reveals nothing about his/her input but the sum value.

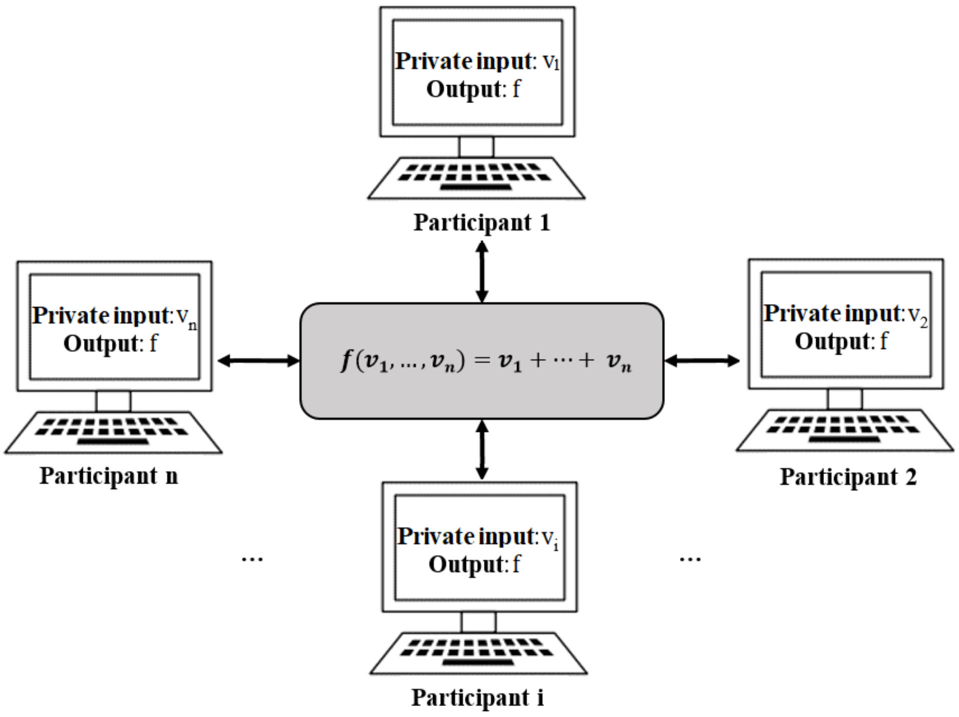


Figure 1.7: The computational model of the secure multi-party sum computation problem

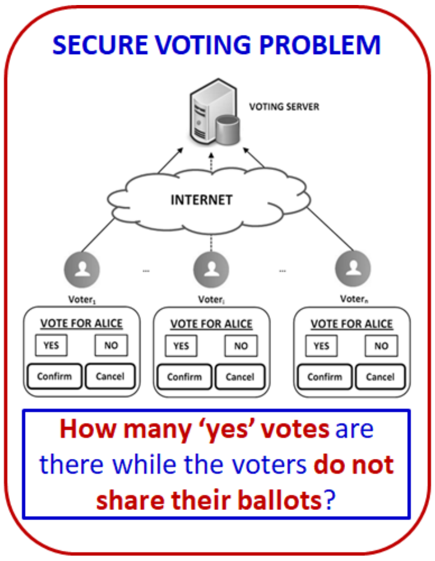
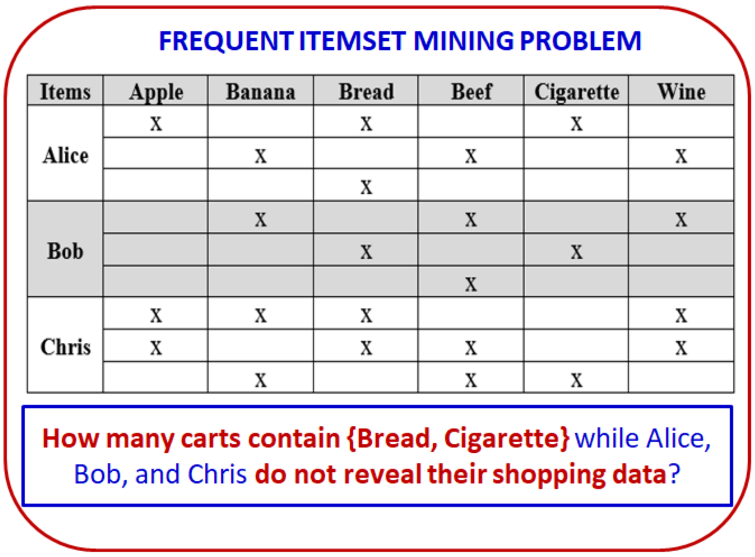


Figure 1.8: The single-candidate end to end decentralized e-voting model

Similarly to general SMC problems, the birth of SMS one has been based on the security requirements of several specific distributed computing tasks. Considering a very classical cryptography task depicted in Figure 1.8 that is the secure e-voting problem where the vote counter needs to compute the sum of 'yes' votes while each voter still privately keeps his/her ballot (i.e. 'yes' or 'no' selection). It is clear that this task is equal to the SMS problem.


FREQUENT ITEMSET MINING PROBLEM

Items	Apple	Banana	Bread	Beef	Cigarette	Wine
Alice	X		X		X	
		X		X		X
			X			
Bob		X		X		X
			X		X	
				X		
Chris	X	X	X			X
	X		X	X		X
		X		X	X	

How many carts contain {Bread, Cigarette} while Alice, Bob, and Chris do not reveal their shopping data?

Figure 1.9: An example of the privacy-preserving frequent itemset mining problem

Another distributed computing task as presented in Figure 1.9 related to the SMS problem is to mine frequent itemset from a large united transaction dataset (e.g. shopping carts), in which each customer reveals nothing his/her data. More precisely, for each itemset, the miner must count the sum of carts containing it while all customers do not want to share their shopping data with anyone.

Let us regard the privacy-preserving Naive Bayes classifier in the horizontal data model (e.g. [23, 33]). To predict the label of a new instance $A = (a_1, ..., a_m)$ based on the multiple users' data records, the miner must and all data users jointly compute the sum of users whose class label is $L^{(i)}$ in which each label $L^{(i)}$ belongs to the set of labels L. Concurrently, the miner also needs to calculate the sum of users whose j^{th} attribute is a_j and class label is $L^{(i)}$. All of the sum values are used for computing probabilistic values $p(L^{(i)}) \prod_{j=1}^{m} p(a_j|L^{(i)})$ to decide the predicted label of the instance A that has the maximum probability. Hence, the privacy-preserving Naive Bayes classification problem in the horizontally distributed scenario is close to the SMS one.

It can be stated that a lot of practical distributed computing tasks have related to the SMS problem. Thus, SMS protocols have been currently applied to various practical computing tasks, such as privacy-preserving recommendation system [21], privacy-preserving data analytics [22], secure e-voting system [12, 13], privacy-preserving classification [23], privacy-preserving association rule mining [6, 7], secure data collection for the smart grid [24], and secure auction [25, 26].

1.2.2. Related work

In the literature, SMS problem has attracted a lot of attention from researchers. Up to now, SMS protocols have been based on two approaches: *non-cryptographic* and *cryptographic* ones. In this section, the typical SMS protocols following these approaches are comprehensively reviewed about both the security and performance properties. For convenience, it is assumed that there are n parties joining a SMS protocol execution, in which the i^{th} party and his/her private input value are correspondingly denoted as U_i and v_i.

1.2.2.1. Review of typical `SMS` *protocols*

(i) The non-cryptographic approach

`SMS` protocols based on the non-cryptographic approach often require each party to split his/her private value into several parts and then share them with others though secure communication channels. A number of such typical `SMS` protocols are reviewed as follows.

It is widely known that the first `SMS` protocol based on the non-cryptographic approach was introduced in [44]. Lately, the improved variant of this protocol was presented in [6] by Clifton et al. Basically, each user U_i of these protocols hides his/her private value v_i by adding it to the number received from the user U_{i-1}, then sharing the result for the user U_{i+1}. Hence, the cost of the protocols [6, 44] is inexpensive, but the private value of U_i is revealed if U_{i-1} and U_{i+1} collude together. In the other words, these protocols have good performance but low level of security.

Urabe et al. [7] proposed a highly secure sum protocol solving privacy-preserving association rules mining problem. In this `SMS` protocol, excepting the special party U_0, each party U_i of this protocol separates his/her private value v_i into $(n-i)$ parts $\{v_{i,i}, v_{i,i+1}, ..., v_{i,n-1}\}$, after that he/she keeps $v_{i,i}$ and shares $\{v_{i,i+1}, ..., v_{i,n-1}\}$ for the parties $\{U_{i+1}, ..., U_{n-1}\}$, respectively. Thus, this protocol may prevent $(n-2)$ corrupted users, but its communication cost is relatively high. Additionally, in the case of large number of parties, it is quite expensive and impractical to establish communication channel between each pair of participants.

Zhu et al. [8] presented a collusion-resisting secure sum protocol, in which the private number v_i of the party U_i is masked in the phase 1 of this protocol. In particular, each participant U_i randomly chooses t different random numbers $\{v_{i\_1}, v_{i\_2}, ..., v_{i\_t}\}$ (t is a given constant positive integer), then shares them for t different others who are randomly chosen by himself/herself. Continuously, the party P_i hides his/her private number v_i by adding or subtracting v_i to the values received from others. As a result, it can be seen that the privacy and execution cost of each party U_i depends on the number t. More specifically, if t is small, then the communication cost of U_i is inexpensive, but the private number v_i can be easily learned, and otherwise. In the other words, the protocol [8] must suffer from the trade-off between the security and performance properties.

Zhang et al. [45] propounded a `SMS` protocol called the rational secure sum one. At the first step of this protocol, each party U_i randomly chooses $(n-1)$ different integers $\{r_i^1, r_i^2, ..., r_i^{i-1}, r_i^{i+1}, ..., r_i^n\}$ and correspondingly sends them to the others $\{U_1, U_2, ..., U_{i-1}, U_{i+1}, ..., U_n\}$. The party U_i then adds his/her private value s_i to all values r_i^j $(j = 1, 2, ..., i-1, i+1, ..., n)$ to the value v_i. In the second step of the protocol [45], each party U_i subtracts all values r_j^i (received from others) from the value v_i. Hence, the protocol of Zhang et al. [45] has the capability to prevent $(n-2)$ colluding parties. Moreover, differently from existing traditional `SMS` protocols, each party U_i of the protocol [45] obtains the sum with complete fairness by executing the function *GenarateTag* in the end step. In fact, the fairness property can be crucial for several `SMC` protocols in some cases (e.g. the case of contract signing [1]), but it is unessential to guarantee this property in many contexts. For example, in the case of credit scoring problem that the miner cooperates with the bank customers to compute the total number of good-rank customers, it does not make sense to share the results with the bank customers. Besides, because each party U_i must transfer messages with $(n-1)$ others, the protocol of Zhang et al. [45] has the same disadvantages with that of Urabe et al. [7].

Li et al. [21] propounded an unsynchronized `SMS` protocol that was applied to a privacy-preserving collaborative filtering problem. In this protocol, each participant separates his/her secret value into t parts, then securely shares them to t online participants randomly chosen by himself/herself. Clearly, if t online participants collude together, then each participant's secret value is revealed. Consequently, the protocol of Li et al. also has a trade-off between the security and performance.

Croce et al. [24] proposed a secure sum computation (SSC) tool as a building block of privacy-preserving overgrid scheme used for securely collecting data in the smart grid. In particular, the SSC tool [24] privately sums the secrets of n distributed nodes by requiring the nodes to executing a protocol that is similar to the previous one [6]. As a result, the secure sum protocol of Croce et al. [24] is only suitable for applications having weak-security constraints.

Based on the same idea of the protocol [7], Luo et al. [46] improved the secure multi-party sum computation protocol to resist clients dropping out. However, this new protocol also requires all participants to communicate together for transferring messages that brings big inconvenience to distributed computation models.

(ii) The cryptographic approach

In contrast, `SMS` protocols based on the cryptography field use homomorphic cryptosystems such as ElGamal cryptosystem [27] or Paillier encryption [28] to securely compute the sum value while still protecting each participant's private input. Next, the `SMS` protocols following this approach are reviewed.

Xun Yi and Yanchun Zhang [47] employed two semi-trusted mixers (denoted as Mixer 1 and Mixer 2) to construct a secure protocol for computing sum of counts that is used to build privacy-preserving Naive Bayes classifiers. This protocol then is improved to compute a series of sum values by encrypting multiple inputs in one ciphertext. To obtain each sum value, the protocol [47] requires that each user U_i divides his/her private count v_i into two parts in which the first and second parts are encrypted by the Paillier public keys of Mixer 1 and Mixer 2, then shares these ciphertexts for Mixer 1 and Mixer 2, respectively. At the end step of the protocol, the semi-trusted mixers can obtain the sum of counts by aggregating all ciphers received from users and jointly running the two-party protocol. It can be seen that if the two mixers conclude, then each user's count is disclosed. In the other words, the protocol [47] has low level of security.

In 2011, Shi et al. [48] proposed a `SMS` protocol that allows the aggregator computes the sum of all parties' private inputs without disclosing these values. To obtain this goal, each party U_i's private input v_i is encrypted into the ciphertext $g^{v_i}.H(t)^{sk_i}$, in which g is a generator, $H(.)$ is a secure hash function, t is time step, and sk_i is U_i's secret key chosen by a trusted dealer. The aggregator recovers the sum value by multiplying all ciphertexts, then executing the brute-force search or Pollard's lambda method. It is not hard to see that the security of the protocol [48] is weak, because of using the trusted party.

Jung et al. [49] propounded a collusion-tolerable privacy-preserving sum without secure channel. Before submitting to the aggregator, the party U_i converts his/her private value v_i into the ciphertext $C_i = (1 + p.v_i).(\frac{g^{r_{i+1}}}{g^{r_{i-1}}})^{r_i} \mod p^2$ where p, g is the public cryptographic parameters, $g^{r_{i+1}}, g^{r_{i-1}}$ are the corresponding public keys of U_{i+1}, U_{i-1}, and r_i is the private key of U_i. After receiving the ciphertexts from all participants, the aggregator calculates $C = \prod_{i=1}^{n} C_i \mod p^2$, then efficiently computes the final sum by exploiting the modular property via the equation $\sum_{i=1}^{n} v_i = \frac{C-1}{p}$. Unfortunately, Datta and Joye pointed out in [50] that the private value v_i of the party U_i is easily recovered by anyone from the ciphertext C_i as $v_i = \frac{1-C_i^{p-1} \mod p^2}{p} \mod p$. Hence, the SMS protocol of Jung et al. [49] has low level of security.

Having the same idea to the privacy-preserving frequency mining protocol in [33] (see more detail in the next section), Badsha et al. [51] proposed a SMS protocol. After that, the authors of [51] used this SMS protocol to construct a solution for a practical privacy-preserving recommendation system. To get the similarity used for generating recommendations for the target user, Badsha et al. securely compute mediate sum values by performing a SMS protocol that requires each user U_i transforms his/her input, e.g. the rating of U_i on the j^{th} item $r_{i,j}$, into the ciphertext of ElGamal encryption $E(r_{i,j}) = (g^{r_{i,j}}.Y^{r_i}, g^{r_i})$, in which g is a generator, r_i is the private key of U_i, and Y is the global public key computed from all users' local public keys (i.e. g^{x_i}, $i = 1, ..., n$). Because of the properties of ElGamal cryptosystem, the protocol [51] can correctly compute the necessary sums as well as privately protecting each user's input values. However, its performance is quite poor, since all participants (including both the users and server) must execute up to three rounds of computation.

Based on a random shuffle function and the ElGamal encryption, Mehnaz et al. [22] proposed a collusion-resisting SMS protocol applied to privacy-preserving regression and classification techniques. In the first phase of this protocol, each party U_i first separates his/her private value v_i into s segments $\{v_{i_1}, v_{i_2}, ..., v_{i_s}\}$, then U_i sequentially encrypts these s values by using the ElGamal cryptosystem public keys of the mediator, $U_1, U_2, ..., U_n$, orderly. Continuously, each party U_i sends the $(n+1)$ layers of encryption-ciphertexts to the mediator. In the second phase of the protocol, the mediator shares all $(n*s)$ ciphertexts for the party U_n. Next, U_n strips off one layer of encryption for $(n*s)$ ciphertexts, then randomly re-orders the results to share for U_{n-1}. Similarly, the parties $U_{n-1}, U_{n-2}, ..., U_1$ do this until n layers of encryption of $(n*s)$ ciphertexts are stripped and no one knows that each of $(n*s)$ outputs is the ciphertext (under the mediator's public key) of which party's segment. At the end of the protocol, the mediator decrypts $(n*s)$ outputs into $(n*s)$ segments, then computes the sum of these values to obtain the global sum. It is easy to see that the ElGamal encryption and the random shuffle function securely protect the private input of each party, as well as correctly computing the sum value. Nevertheless, each party of the protocol [22] is required to operate too many times of ElGamal encryption and decryption tasks. Consequently, the computational complexity of the protocol [22] is extremely high. Moreover, this protocol must establish communication channels between U_i and U_{i+1} $(i = 1, 2, ..., n-1)$.

With the aim of proposing a differential private Naive Bayes classifier, Li et al. constructed a `SMS` function in the study [23]. In this proposal, to help the data receiver to obtain the classification model, the data collector cooperates with the data providers to compute the sum of counts. In more detail, first of all, an honest dealer called the system initializes cryptographic systems and generates some public and private parameters for the entities. Next, each data provider's local counts are encrypted by the Paillier encryption public keys of the data collector and the data receiver, then the ciphertexts are submitted to the data collector for this entity to aggregate and share the results for the data receiver. Finally, based on the homomorphic property of Paillier cryptosystem and the Laplace mechanism, the data receiver extracts the differential model of a NB classifier. It is widely known that the Paillier cryptosystem has high computational cost, so the `SMS` function of Li et al. [23] has poor performance. More seriously, the private data of providers completely depends on the honest dealer.

To develop a secure decentralized training technique for privacy-preserving deep learning models, Tran et al. [52] proposed an efficient and secure sum protocol enabling a large group of parties to jointly compute a sum of private inputs. The new solution of Tran et al. can work not only with integer number, but also with floating point number. To do this, Tran et al.'s protocol uses the original one [33], as well as defining a function $f : \mathbb{Z}_p \times \mathbb{R} \mapsto \mathbb{R}$. Next, each party U_i encrypts his/her private value v_i into two ciphertexts $E_i = f(\frac{X^{y_i}}{Y^{x_i}}, k^{v_i}) = \frac{X^{y_i}}{Y^{x_i}}.k^{v_i}, F_i = \frac{X^{y_i}}{Y^{x_i}}.g^{t_i}$, then sends them to the master party. By computing $T = \prod_{i=1}^{n} E_i, S = \prod_{i=1}^{n} F_i$ and solving logarithm problems over the field $\mathbb{R}$, the master party outputs the sum $\sum_{i=1}^{n} v_i$. It is not hard to see that if the number of parties n is large, then T, S are extremely big integers and the issue of solving logarithm problems over the field $\mathbb{R}$ is completely expensive. Consequently, the protocol of Tran et al. [52] has poor performance.

Recently, the SMS protocols [53–56] based quantum computing techniques have been considered as a new interesting research topic in the SMC field. Although such protocols can withstand potential code-breaking attempts by quantum computers, they need to be supported by a trusted third party and their computational cost is high.

1.2.2.2. Review of typical privacy-preserving frequency computation protocols

In essence, privacy-preserving frequency computation (PPFC) is considered as a special case of secure multi-party sum computation problem, because PPFC has the same objective with SMS ones, but the private value v_i of each party U_i is only ”0” or ”1”. Up to now, a lot of secure protocols have been proposed for computing frequency value. They have been used to construct practical problems such as privacy-preserving ID3 tree and association rule mining [33], privacy-preserving Naive Bayes classifier [23,57], secure electronic voting system [12,13,58]. Typical PPFC protocols are reviewed as follows.

Yang et al. introduced a PPFC protocol in [33]. To securely compute a frequency value, this protocol requires each party U_i to encrypt his/her private input v_i (using two global public keys derived from all parties' public keys) into up to two ciphertexts m_i, h_i and submit them to the miner. When receiving the parties' ciphertexts, the miner multiplies all of them to get the value K and runs the brute-force algorithm to obtain the frequency value form K. Because of the properties of ElGamal encryption, this protocol strongly protects each user's privacy without loss of accuracy. In addition, it requires no communication channel between each pair of data users as well as multi-round interaction between the miner and each data user. However, the computational cost of the miner is quite expensive, especially in the case that the number of parties is large or the protocol [33] is executed multiple times.

In 2009, Wu et al. [57] proposed a PPFC protocol for accurate and private mining of support counts in fully distributed scenario. This protocol and that of Yang et al. have the same method of computation, but differently from [33], the protocol [57] saves the overhead for certificating the parties' public keys by employing Boneh–Franklin identity-based encryption scheme. Nevertheless, the security of the protocol [57] depends on the entity who creates a tuple of system parameters. Thus, this protocol has low level of security.

Inspiring from the work [59] (*Hao et al., 2006*), *Hao et al., 2010* [12] propounded a PPFC protocol that is used for an anonymous voting scheme to securely compute the candidate's overall total while revealing nothing about each voter's ballot. To do this, the protocol [12] requires each voter U_i to transform his/her ballot v_i ("0" or "1" value) into a ciphertext of ElGamal encryption $g^{v_i}(g^{y_i})^{x_i}$ using his/her private key x_i and the reconstructed public key g^{y_i} derived from other $(n-1)$ voters' public keys. Next, to obtain the voting result, the counter aggregates all ciphertexts received from the voters, then runs the brute-force or Shanks' baby-step giant-step algorithm. Similarly to the protocol of Yang et al. [33], each voter's ballot is privately protected, and the voting result is exactly ensured. However, because the voters' public keys are shared for all and each voter needs to compute his/her reconstructed public key from $(n-1)$ public keys by himself/herself, it can be seen that the protocol [12]'s communication cost is high and each voter's computational cost is expensive.

Improved from the work [12], a variant of PPFC protocol was presented to construct a cryptographic e-voting protocol called DRE-i [13]. Instead of using individual devices (e.g. personal computer, mobile phone) to compute the reconstructed public key g^{y_i} and the ballot as in the original protocol [12], each voter U_i of the protocol [13] only interacts with the machine equipped a touch-screen. In addition, this protocol also assumes the existence of a system entity who computes the reconstructed public keys for all voter. As a result, the security of the voting system is guaranteed and each voter's computational complexity and the system's communication cost are greatly reduced, but the system entity's computational complexity is high.

In 2018, *Hao et al., 2018* developed an elliptic curve analog of ElGamal system-based version of [13] that was then applied to a verifiable classroom voting system [58]. Because of the elliptic curve analog of ElGamal system's advantages, the communication cost and computational complexity of each voter is optimized in [58], but the voting system's total computational complexity is equal to that of the protocol [13].

1.2.2.3. Review of typical secure multi-sum computation protocols

In essence, the aim of the secure multi-sum computation problem is to simultaneously compute multiple sum values in one round of computation. This problem is also called the secure aggregation one [60–63] that is to compute sum of private vectors over multiple data sources within privacy constraints.

In fact, this problem is quite popular. For example, secure multi-candidate election systems (e.g. [12, 64]), the voters want to compute each candidate's overall total while they do not reveal their ballots (i.e. yes or no selection). Another example of the secure multi-sum computation problem is privacy-preserving Naive Bayes classifier for the horizontal data setting (e.g. [47]) that requires the miner to cooperate with the voters to privately compute multiple frequency values used for calculating the necessary probabilities. In addition, there are a lot of other practical privacy-preservation problems related to the secure multi-sum computation problem, such as privacy-preserving linear regression [35], privacy-preserving logistic regression [65]).

The direct way to securely compute multiple sum values is to execute secure multi-party sum or privacy-preserving frequency computation protocols in the studies [7, 12, 33] or the third publication multiple times. However, this method requires high communication and computational costs.

Several other studies [12, 47, 64–66] proposed improved solutions, in which many sum values can be simultaneously calculated by packing multiple input values in a unique ciphertext. However, the solutions [47, 64, 66] based on the Paillier encryption and the ones using the LWE-based encryption scheme (e.g. [35]) are unsuitable for privacy-preservation issues requiring high level of security, because they need a trusted party to generate security parameters. For the proposals based on the ElGamal encryption (e.g. [12]), their computational cost is expensive, since logarithm discrete problems with large scale of solutions must be solved.

To construct privacy preserving training of Naive Bayes models, Kjamilji et al. propounded a secure sum protocol in [67] using an aggregation cloud server (TACS), an encryption/decryption server (EDS) working with dataset owners. The most serious problem of Kjamilji et al.'s protocol is to assume that there is no collision between the TACS and EDS in any scenario. Clearly, this assumption is perfectly weak.

To build privacy-preserving training models for federated learning, Kairouz et al. [68] and Chen et al. [62] developed secure noise-sum values computation methods by adding noise before performing secure aggregation. As a result, the propotocols must have a trade-off betwen the privacy and correctness.

By using several popular cryptographic primitives, such as Shamir's t-out-of-n secret sharing technique, symmetric authenticated encryption public key infrastructure, Bonawitz et al. [60] presented two variants of practical secure aggregation protocol that have a constant number of rounds, low communication overhead, robustness to failures. However, this proposal cannot preserve honest parties' privacy in the case the server colludes with more than $\lceil \frac{n}{3} \rceil$ (n is the number of participants). Additionally, the secure aggregation protocols in [60] are not suitable for large models, because they requires pairs of users to communicate together.

Based on the idea of [60], Bell et al. [61] proposed a complete communication graph used for the parties transferring messages to obtain the efficiency property while maintaining the security guarantees. Nevertheless, similarly to the protocols in [60], each pairs of users in [61] still needs to communicate together.

With the aim of enhancing the efficiency of the protocol in [61] and enabling input validation function based on zero-knowledge proof techniques, the authors of [63] introduced three new extended secure aggregation protocols. In the first one, Bell et al. [63] used the ring learning-with-errors (RLWE)-based encrytion that requires quite expensive cost of computation and a trusted party to generate the security parameters. For the ACORN-detect and ACORN-robust protocols, the capability to validate the constraints of input values is not perfectly novel, because it has been provided by several previous studies [12].

To prevent deletion and tampering attacks from aggregators, Jianhong Zhang and Chenghe Dong [69] propounded a privacy-preserving data aggregation scheme based on the idea of the protocol [12]. Unfortunately, the protocol [69] must use a trusted authority who is responsible for initializing system parameters for all in a trusted environment.

1.2.2.3. Summary of existing secure multi-party sum computation protocols

It can be seen that the non-cryptographic SMS protocols often have low computational complexity, but they must suffer a trade-off between security and communication cost. In addition, they are unsuitable for multi-party computational models with a large number of participants.

In contrast, SMS protocols based on cryptography can obtain high level of security, and such protocols have been preferred. However, the cryptographic approach-based SMS protocols often have pricey cost of computation.

For the cases simultaneously computing multiple sum values, the existing secure multi-sum computation protocols still have serious drawbacks, such as low level of privacy, high computational cost or poor applicability in large models.

Therefore, it is necessary and significant to design SMS protocols that should be not only secure against malicious adversary but also efficient in real-life applications.

1.3. Conclusion

In this chapter, the thesis has represented background and preliminaries in the SMC field. The thesis then formulated the SMS problem and pointed its importance in practice. To find out potential research issues for the SMS problem, the related work has been fully analyzed.

CHAPTER 2. PROPOSING EFFICIENT SECURE MULTI-PARTY SUM COMPUTATION PROTOCOLS

This chapter meticulously typical secure multi-party sum computation protocols. Based on the analysis result, this chapter proposed three new secure multi-party sum computation protocols having both high security level and good performance.

2.1. Analysis of typical secure multi-party sum computation protocols

This section fully analyzes the most popular SMS protocols related to the thesis, i.e. the simple secure sum protocol of Schneier et al. [44], the SMS protocol of Urabe et al. [7], a series of secure sum protocols of Hao et al. [12, 13, 58] used in electronic voting systems, and especially the privacy-preserving frequency computation protocol of Yang et al. [33] that most of this thesis's proposals are inspirited from. For each of these typical protocols, its security and performance properties will be comprehensively evaluated. Based on this analysis, the research issues of this thesis are explicitly determined.

It is also recalled that there are n parties $\{U_1, U_2, ..., U_n\}$ where the i^{th} party U_i owns a private value v_i $(i = 1, 2, ..., n)$. The aim of SMS protocols is to correctly compute the sum value $V = \sum_{i=1}^{n} v_i$ while all parties do not reveal their private values.

For convenience, this section uniformly uses the above notations in all analysis.

2.1.1. Simple secure multi-party sum computation protocol

2.1.1.1. The main phases

As mentioned before, the early SMS protocol was originally introduced in [44]. This protocol assumes that U_1 is chosen as the master party computing the final sum value for others. The main phases of the simple SMS protocol [44] is presented in Protocol 2.1.

Protocol 2.1: The simple secure multi-party sum computation protocol of Schneier et al. [44]

Phase 1: Party U_1 does

- Chooses a random number r in the same domain with the values v_i and V
- Computes $d = r + v_1$
- Sends d to U_2 via secure channel

Phase 2: Party U_i ($i = 2, ..., n$) does

- Updates $d = d + v_i$
- Sends d to U_{i+1} (if $i = 2, ..., n-1$) or U_1 (if $i = n$) via secure channel

Phase 3: Party U_1 does

- Computes $V = d - r$
- Broadcasts V to others

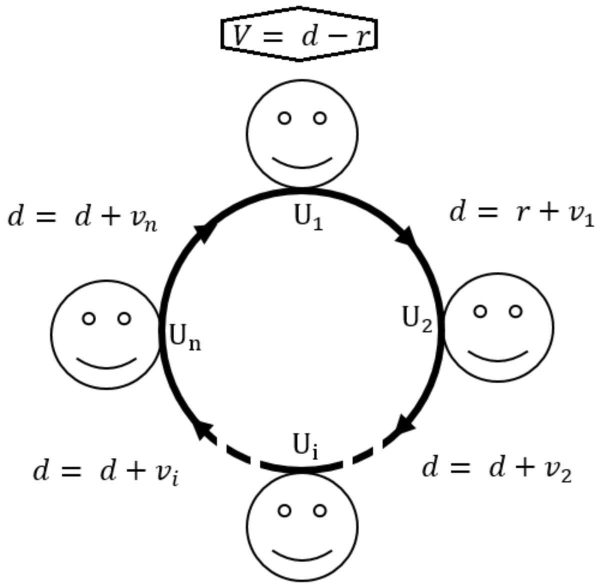


Figure 2.1: The computational model of the simple secure multi-party sum computation protocol

2.2.1.2. Security analysis

i. Proof of correctness

It was proved that the output V of Protocol 2.1 is the sum of all input values v_i's.

Indeed, considering the phases 1 and 2, it can be seen that $d = r + v_1 + v_2 + ... + v_n$. Moreover, because $V = d - r$ in the phase 3, $V = v_1 + v_2 + ... + v_n$.

ii. Privacy analysis

It is understandable that every party U_i's private value v_i is masked by the number that received from U_{i-1} ($i = 2, 3, ..., n$). Hence, if U_{i-1} collude with U_{i+1} who is received U_i's message, then the party U_i's private value v_i is disclosed. Thus, the security of the protocol [44] is quite weak.

2.1.1.3. Performance analysis

In the protocol [44], all of parties $\{U_1, U_2, ..., U_n\}$ cooperatively construct a ring communication network (see in Protocol 2.1). Through this model, each user only sends and receives one message using secure channels. As a result, this protocol requires low cost of computation and communication. In other words, the protocol [44] has high performance. However, it is quite complex to establish a ring communication network in the case of large number of parties.

2.1.2. Secure multi-party sum computation protocol of Urabe et al.

2.1.2.1. The main phases

As shown in the first chapter, the main idea of the protocol of Urabe et al. [7] is to each party splits his/her input value into several parts, then shares them for other parties. Similarly to the protocol [44], U_1 is the master party computing the sum value. This protocol is briefly described in Protocol 2.2.

Protocol 2.2: Urabe et al.'s protocol [7] for securely computing the multi-party sum value

Phase 1: Each party U_i ($i = 2, ..., n$) does

- Splits v_i into $(n - i + 1)$ random parts $\{v_i^{(i)}, ..., v_i^{(n)}\}$ such that $v_i = \sum_{j=i}^{n} v_i^{(j)}$
- Sends $v_i^{(j)}$ to U_j ($j = i + 1, ..., n$)

Phase 2: Each party U_i ($i = 2, ..., n$) does

- Computes $v_i' = v_i^{(i)} + \sum_{j=1}^{i-1} v_j^{(i)}$
- Sends v_i' to U_1

Phase 3: U_1 does

- Computes $V = v_1 + \sum_{i=2}^{n} v_i'$
- Outputs V

2.1.2.2. Security analysis

i. Proof of correctness

It is shown that the output V of Protocol 2.2 is the sum of all input values v_is. Indeed, the following transformations are performed.

$$\begin{aligned}
v = \ & v_1 + \sum_{i=2}^{n} v_i' \\
= \ & v_1 + \sum_{i=2}^{n} (v_i^{(i)} + \sum_{j=1}^{i-1} v_j^{(i)}) \\
= \ & v_1 + \sum_{i=2}^{n} v_i^{(i)} + \sum_{n=2}^{n} \sum_{j=1}^{i-1} v_j^{(i)} \\
= \ & v_1 + \sum_{i=2}^{n} (v_i^{(i)} + ... + v_i^{(n)}) \\
= \ & v_1 + \sum_{i=2}^{n} v_i \\
= \ & \sum_{i=1}^{n} v_i.
\end{aligned}$$

Thus, the `SMS` protocol of Urabe et al. is correct.

ii. Privacy analysis

Excepting the private value v_1 of U_1 is not shared for any party, the values v_i of the parties U_i $(i = 2,...,n)$ are depended on other $(n-1)$ parties. Hence, Protocol 2.2 can securely protect the privacy of each honest party against $(n-2)$ parties colluding together.

2.1.2.3. Performance analysis

It can be seen in the protocol of Urabe et al. [7] that each party is required to establish secure (both private and authenticated) communication channels with $(n-1)$ others to prevent adversaries from eavesdropping. Furthermore, the number of communication messages used for computing is $\frac{n(n-1)}{2}$. Consequently, the computational and communication costs of the protocol [7] are high. Besides, for the model with a large number of participants, it is impractical to require each tuple of parties to setup a communication channel together. This also makes the applicability of the protocol [7] reduce greatly.

2.1.3. Secure multi-party sum computation protocol of Hao et al., 2010 in an electronic voting system

To design a secure and efficient decentralized-voting solution, *Hao et al., 2010* [12] proposed a secure computation protocol that has the same goal with secure frequency computation protocols. In the electronic voting setting, it is assumed that there are n voters $\{U_1,...,U_n\}$ who need to vote "yes/no" ballots ("$1/0$", respectively) for a candidate. In particular, each U_i votes for the candidate by submitting a private value $v_i \in \{0,1\}$ $(i = \{1,2,...,n\})$. The protocol of *Hao et al., 2010* [12] aims to correctly compute $V = \sum_{i=1}^{n} v_i$ while each voter reveals nothing about his/her v_i. It can be stated that the secure frequency computation protocol of *Hao et al., 2010* [12] is one of the most typical `SMS` protocols, and it has been currently used as an important secure building block for various end-to-end voting systems [58, 70–72].

2.1.3.1. The main phases

In the protocol [12], all participants agree to adopt a cyclic group $\mathbb{G}$ of large prime order q with the generator g such that discrete logarithm problems in $\mathbb{G}$ are hard. The main phases of this protocol is described in Protocol 2.3.

Protocol 2.3: The secure multi-party sum computation protocol [12] of *Hao et al., 2010* for secure e-voting solution

---

Phase 1: Each party U_i ($i = 1, ..., n$) does

- Chooses a private key $x_i \in \{1, ..., q\}$ and computes the public key $X_i = g^{x_i}$
- $U_i \rightarrow$ Server: X_i

Phase 2: Server does

- Server $\rightarrow U_i$: $X_1, X_2, ..., X_n$

Phase 3: Each party U_i ($i = 1, ..., n$) does

- Computes $c_i = g^{v_i}.(\frac{\prod_{j=1}^{i-1} X_j}{\prod_{j=i+1}^{n} X_j})^{x_i}$
- $U_i \rightarrow$ Server: c_i

Phase 4: Server does

- Computes $K = \prod_{i=1}^{n} c_i$
- Runs the Shanks' algorithm to find out $V \in \{0, ..., n\}$ satisfying $g^V = K$

2.1.3.2. Security analysis

i. Proof of correctness

It needs to be shown that if the server finds out a value $V \in \{0, ..., n\}$ that satisfies $g^V = K$, then V is correct.

Assume that $g^V = K$. Then:

$$\begin{aligned}
g^V &= K \\
&= \prod_{i=1}^{n} c_i \\
&= \prod_{i=1}^{n} g^{v_i} \left(\frac{\prod_{j=1}^{i-1} X_j}{\prod_{j=i+1}^{n} X_j}\right)^{x_i} \\
&= \prod_{i=1}^{n} g^{v_i} \left(\frac{\prod_{j=1}^{i-1} g^{x_j}}{\prod_{j=i+1}^{n} g^{x_j}}\right)^{x_i} \\
&= \prod_{i=1}^{n} g^{v_i} \prod_{i=1}^{n} \left(\frac{\prod_{j<i} g^{x_j}}{\prod_{j>i} g^{x_j}}\right)^{x_i} \\
&= g^{\sum_{i=1}^{n} v_i}
\end{aligned}$$

Thus $g^V = g^{\sum_{i=1}^{n} v_i}$, and therefore $V = \sum\limits_{i=1}^{n} v_i$.

ii. Privacy analysis

Considering phases 1 and 2 of the protocol presented in Protocol 2.3, each voter U_i sends $(g^{x_i}, g^{v_i}(\frac{\prod_{j=1}^{i-1} X_j}{\prod_{j=i+1}^{n} X_j})^{x_i})$ to the server. It is clear that this tuple is a ciphertext of the ElGamal encryption (g^{x_i}, MY^{x_i}) where $M = g^{v_i}$, the public key is $Y = (\frac{\prod_{j=1}^{i-1} X_j}{\prod_{j=i+1}^{n} X_j})$, and x_i is randomly chosen from $\{1, ..., q\}$. Hence, the private value v_i is securely protected against the server and the corrupted voters.

2.1.3.3. Performance analysis

i. Computational complexity

For the server, this entity only performs $(n-1)$ modular multiplication operations and runs Shanks' algorithm to find out V. As a result, the computational complexity required for the server is low.

Regarding the computational complexity of the voters, each U_i needs to execute 2 modular exponentiation, n modular multiplication, and 1 modular multiplicative inverse operations. Hence, in the case of the large number of voters n, each voter of the protocol [12] must spend high cost performing his/her tasks.

ii. Communication cost

As described in Protocol 2.3, each voter only sends two messages to the server, while the server must share the public keys X_i $(i = 1, 2, ..., n)$ for all voters. Thus, the total messages transferred in the protocol [12] is quite large (i.e. $(n^2 + n)$).

2.1.3.4. Variants of the protocol of Hao et al., 2010

To decrease each voter's computational complexity, an improved protocol was proposed for the DRE-based electronic voting system [13] (*Hao et al., 2014*). The most difference between this protocol with the original work of *Hao et al., 2010* [12] is that the server computes the reconstructed key $P_i = \frac{\prod_{j<i} X_j}{\prod_{j>i} X_j}$ for each party U_i (see Protocol 2.4). Hence, each voter only performs 2 modular exponentiation and 1 modular multiplication operations, but the server needs to execute up to $(3n - 4)$ more modular multiplication operations and 1 modular multiplicative inverse operation. Thus, if the number of voters n is large, then the computational complexity of the server is expensive.

It is also noted in the second phase presented in Protocol 2.4 that the sever has to share the specific reconstructed key P_i for each voter U_i $(i = 1, 2, ..., n)$. This means it is complex to the sever ensures the integrity of n reconstructed keys (by creating n corresponding digital signatures or establishing secure/authenticated channels with n voters). Clearly, the computational complexity of the server is also expensive in the case of large number of voters.

Protocol 2.4: The secure multi-party sum computation protocol [13] of *Hao et al., 2014* for secure e-voting solution

Phase 1: Each party U_i ($i = 1, ..., n$) does

- Chooses a private key $x_i \in \{1, ..., q\}$ and computes the public key $X_i = g^{x_i}$
- $U_i \rightarrow$ Server: X_i

Phase 2: Server does

- Computes the reconstructed key $P_i = \frac{\prod_{j=1}^{i-1} X_j}{\prod_{j=i+1}^{n} X_j}$ for each party U_i ($i = 1, ..., n$)
- Server $\rightarrow U_i$: P_i ($i = 1, ..., n$)

Phase 3: Each party U_i ($i = 1, ..., n$) does

- Computes $c_i = g^{v_i}.P_i^{x_i}$
- $U_i \rightarrow$ Server: c_i

Phase 4: Server does

- Computes $K = \prod_{i=1}^{n} c_i$
- Runs the Shanks' algorithm to find out $V \in \{0, ..., n\}$ satisfying $g^V = K$

Moreover, there exists another variant of the protocol of *Hao et al., 2010* in the end-to-end verifiable classroom voting system [58] (*Hao et al., 2018*), where the computations of this protocol are performed over an elliptic curve. As a result, when compared with the protocol in [13] (*Hao et al., 2014*), the one in [58] (*Hao et al., 2018*) has lower communication cost, but its computational complexity is equivalent to the protocol in [13] (*Hao et al., 2014*).

2.1.4. Privacy-preserving frequency computation protocol of Yang et al.

Let $\{U_1, ..., U_n\}$ be n participants where each U_i keeps a private value $v_i \in \{0, 1\}$ with $i = \{1, 2, ..., n\}$. A privacy-preserving frequency computation protocol in the fully distributed setting aims to correctly compute $V = \sum_{i=1}^{n} v_i$ without disclosing the input values v_i's. Clearly, this is a special variant of SMS problem.

2.1.4.1. The main phases

To securely protect the private values of n parties, Yang et al. employed the homomorphic ElGamal encryption. Before the protocol starts, all participants agree to adopt a cyclic group $\mathbb{G}$ of large prime order q with the generator g such that discrete logarithm problems in $\mathbb{G}$ are hard. All private keys are chosen in $\{1,...,q-1\}$ and all computational operations are taken in $\mathbb{G}$. The major steps of Yang et al.'s protocol [33] is presented in Protocol 2.5.

Protocol 2.5: Yang et al.'s protocol [33] for privately computing the frequency value

Phase 1: Each party U_i does

- Chooses two private key x_i, y_i and computes the public keys $X_i = g^{x_i}, Y_i = g^{y_i}$
- $U_i \rightarrow Miner : X_i, Y_i$

Phase 2: $Miner$ **does**

- Computes $X = \prod_{i=1}^{n} X_i, Y = \prod_{i=1}^{n} Y_i$
- $Miner \rightarrow U_i : X, Y$

Phase 3: Each party U_i does

- Computes $m_i = g^{v_i} X^{y_i}, h_i = Y^{x_i}$
- $U_i \rightarrow Miner : m_i, h_i$

Phase 4: $Miner$ **does**

- Computes $K = \prod_{i=1}^{n} \frac{m_i}{h_i}$
- Runs the brute force algorithm to output $V \in \{0,...,n\}$ satisfying $g^V = K$

2.1.4.2. Security analysis

i. Proof of correctness

It needs to be shown that if the miner finds out a value V satisfying the equation $g^V = K$, then V is the sum of all parties' private values.

Assume that $g^V = K$. Then:

$$\begin{aligned}
g^V &= K \\
&= \prod_{i=1}^{n} P_i \\
&= \prod_{i=1}^{n} \frac{g^{v_i}.X^{y_i}}{Y^{x_i}} \\
&= g^{\sum_{i=1}^{n} v_i} \prod_{i=1}^{n} \frac{X^{y_i}}{Y^{x_i}} \\
&= g^{\sum_{i=1}^{n} v_i} \prod_{i=1}^{n} \frac{(\prod_{j=1}^{n} X_j)^{y_i}}{(\prod_{j=1}^{n} Y_j)^{x_i}} \\
&= g^{\sum_{i=1}^{n} v_i} \prod_{i=1}^{n} \frac{(g^{\sum_{j=1}^{n} x_j})^{y_i}}{(g^{\sum_{j=1}^{n} y_j})^{x_i}} \\
&= g^{\sum_{i=1}^{n} v_i} \frac{g^{\sum_{j=1}^{n} x_j \sum_{i=1}^{n} y_i}}{g^{\sum_{j=1}^{n} y_j \sum_{i=1}^{n} x_i}} \\
&= g^{\sum_{i=1}^{n} v_i}
\end{aligned}$$

Hence, $g^V = g^{\sum_{i=1}^{n} v_i}$, thus $V = \sum\limits_{i=1}^{n} v_i$.

ii. Privacy analysis

Without loss of the generality, it is supposed that U_1 and U_2 do not collude and $I = \{3, 4, ..., n\}$. Based on the output V, knowledge of the corrupted parties, the public keys and a few tuples of ElGamal encryption, the algorithm $\mathbb{M}$ simulates to compute m_1, h_1, m_2, h_2 as follows:

$$m_1' = (g^{v_1} g^{x_1 \cdot y_1})(g^{v_1} g^{x_2 y_1}) Y_1^{\sum_{i\in I} x_i}, h_1' = \frac{(g^{v_1} g^{x_1 y_1})(g^{v_2} g^{x_1 y_2}) X_1^{\sum_{i\in I} y_i}}{g^{V - \sum_{i\in I} v_i}} \tag{2.1.1}$$

$$m_2' = (g^{v_2} g^{x_1 y_2})(g^{v_2} g^{x_2 y_2}) Y_2^{\sum_{i\in I} x_i}, h_2' = \frac{(g^{v_2} g^{x_2 \cdot y_1})(g^{v_2} g^{x_2 y_2}) X_2^{\sum_{i\in I} y_i}}{g^{V - \sum_{i\in I} v_i}} \tag{2.1.2}$$

Thus, the protocol [33] can securely protect the honest participants' private values, against up to $(n-2)$ corrupted parties.

2.1.4.3. Performance analysis

i. Computational complexity

Considering the protocol described in Protocol 2.5, each party performs 2 modular exponentiation operations for computing his/her public keys X_i, Y_i, and 2 modular exponentiation and 1 modular multiplication operations for preparing the communication messages m_i, h_i. For the miner, this entity needs to totally execute $(4n+V-4)$ modular multiplication and n modular multiplicative inverse operations. Consequently, if the number of parties n is large, the computational cost of the miner will be expensive.

ii. Communication cost

It can be seen in Protocol 2.5 that each party sends $4|p|$ bits to the miner, while the miner needs to transfer $2\ |p|$ bits to all parties. Thus, the total of communication cost of the protocol [33] is $6n|p|$ bits.

2.1.5. Further discussion

Based on the above analysis, it can be seen that the protocols of Yang et al. [33] and Hao et al. [12, 13, 58] are the most remarkable ones among the typical SMS protocols.

Let us compare the computational complexity among three typical SMS protocols. For convenience, the thesis uses the notations including • T_e, T_m, T_i, T_S that are the times for executing a modular exponentiation, a modular multiplication, a modular multiplicative inverse, Shanks' baby-step giant-step algorithm operations, respectively.

First of all, considering the computational complexity of the server/miner presented in the column 2 of Table 2.1, it takes the server of the protocol [12] the smallest computational complexity among three typical SMS ones. This is understandable, because in the phase 2 described in Protocol 2.3, the server of the protocol [12] only forwards the public keys $\{X_1, X_2, ..., X_n\}$ to all parties without perform-

Table 2.1: The brief comparisons of the computational complexity among three typical `SMS` protocols

Protocols	**The server/miner**	**Each party**
Yang et al.'s [33]	$(4n+V-4)T_m+nT_i$	$4T_e+T_m$
Hao et al., 2010's [12]	$(n-1)T_m+T_S$	$2T_e+T_i+nT_m$
Hao et al., 2014's [13]	$(3n-4)T_m+T_i+T_S$	$2T_e+T_m$

ing any computation. Comparing the computational complexity of the server/miner between the protocols [33] and [13], it takes the miner of the protocol [33] up to $(4n+V-4)T_m+nT_i$ to execute his/her tasks while that complexity of the protocol [13] is only $(3n-4)T_m+T_i+T_S$. The reason of this drawback is because the miner must compute n values $\frac{m_i}{h_i}$ $(i=1,2,...,n)$ in the phase 4 of the protocol [33].

Continuously, regarding each party's computational complexity presented in the third column of Table 2.1, it can be seen that each party's computational complexity of the protocol [13] is lower than that of the one [33]. This is that because the protocol [13] only requires each party to use a unique private key (i.e. x_i in Protocol 2.4), while each party of the protocol [33] have to employ 2 private keys (i.e. x_i, y_i in Protocol 2.5). For the protocol [12], although each party of this protocol also uses a unique private key as described in Protocol 2.3, he/she still spends quite high cost executing his/her tasks. This is because he/she has to compute his/her reconstructed key by himself/herself, the server only plays a role as a postman in the phase 2 of the protocol [12].

Although the previous work of Yang et al. [33] had been proposed in 2005, but the above analysis showed that this is the most potential `SMS` protocol among the typical ones. It is easy to understand, because:

(1) In the case that distributed computing problems only require to execute `SMS` protocols once (e.g. the single-candidate secure e-voting problem), the protocol [33] can be improved by requiring each party U_i to compute $P_i = \frac{g^{v_i}X^{y_i}}{Y^{x_i}}$, and send a unique value P_i to the miner in the phase 3. According to this setting, when com-

pared with the original work [33] and the typical SMS protocols [12,13], the improved protocol only requires each party U_i to perform one more modular multiplication operation and its computational cost will increase by a negligible amount, but the total computational cost of the improved protocol will reduce n modular multiplicative inverse operations, the computational cost of the miner of the improved protocol will decrease $(n-1)$ modular multiplication and n modular multiplicative inverse operations, and the communication cost of the improved protocol will reduce n messages. Moreover, in the re-designed protocol, each party U_i only sends a unique value P_i to the miner in the phase 3, so it is possible to integrate an authenticated method into the improved protocol to create an efficient SMS protocol can be directly implemented on public networks (e.g. Internet) without pre-establishing secure/authenticated channels among the participants. Clearly, the above improvement can bring good changes. Besides, the performance of [33] can be additionally boosted by replacing the brute-force algorithm by efficient methods (e.g. Shanks' algorithm [73]) to solve the discrete logarithm problem $g^V = K$ in the phase 4, and employing elliptic curve-based cryptosystems. These ideas are going to be actualized in Section 2.2.1, Section 2.2.2, and Section 3.1 of the thesis.

(2) In the case that computational tasks require to execute SMS protocols multiple times (denoted as ns times), the improved variant of [33] only uses $\left\lceil \frac{1}{2} + \sqrt{2ns + \frac{1}{4}} \right\rceil$ tuples of private and public keys without security concerns while those numbers of the original work [33] and the typical SMS protocols [12,13] are up to $2ns$ and ns, respectively. Especially, for several specific applications that need to simultaneously compute multiple sum values only in one round of computation (e.g. privacy-preserving Naive Bayes classification problem), it is possible to create a secure multi-sum computation protocol by combining the improved protocol and some cryptography techniques. This idea is implemented in Section 2.2.3 and Section 3.2 of the thesis.

2.2. Proposed secure multi-party sum computation protocols

Based on the above ideas, this section proposes three novel SMS protocols having unique features. It is also recalled that all of the new proposals are based on the semi-honest model.

2.2.1. Privacy-preserving frequency computation protocol based on elliptic curve ElGamal cryptosystem

This contribution is related to **Publication** 1. In this section, the thesis propounds a privacy-preserving frequency computation protocol based on an elliptic curve analog of the ElGamal cryptosystem that allows a set of parties to securely compute the sum of all '1' input values over authenticated communication channels.

2.2.1.1. Introduction

As mentioned before, the PPFC problem is a variant of SMS one, and it is originated from practical applications that require to securely computing one or many frequency values (e.g. secure e-voting schema). Currently, the existing PPFC protocols have high cost of computation, or a trade-off between security and computational complexity. As a result, the thesis proposes a new PPFC protocol having both high level of security and good performance.

This contribution of the thesis is based on the idea that consists of two main steps as follows:

(1) First, re-designing the phase 3 of [33] by requiring each party U_i to compute $\frac{g^{v_i} X^{y_i}}{Y^{x_i}}$.

(2) Then, transforming the re-designed protocol into an elliptic curve analog of the ElGamal system-based protocol.

When compared with both the original work [33] and other typical protocols, the improved PPFC protocol has not only high level of security but also good performance and wide applicability.

2.2.1.2. Privacy-preserving frequency computation protocol

i. Problem statement

The PPFC problem in the fully distributed setting is stated as follows. It is supposed n users $\{U_1,...,U_n\}$ where each user U_i owns a private value $v_i \in \{0,1\}$, and the miner who wishes to find out the sum $v = \sum_{i=1}^{n} v_i$. The computational method helps the miner to obtain this goald is called PPFC protocol.

ii. Definition of security

The proposed protocol follows the semi-honest model that each user complies the rules of the protocol, so the following definition of security is derived from the standard definition represented in Section 1.1.

Definition 2.2.1. *Suppose that each party U_i has his/her private keys p_i, q_i and public keys P_i, Q_i. A frequency computation protocol protects each party's privacy against the miner and t corrupted participants in the semi-honest model if, $\forall I \subseteq \{1,...,n\}$ such that $\|I\| = t$, there exists a probabilistic polynomial-time algorithm $\mathbb{M}$ such that:*

$$\{\mathbb{M}(v, [v_i, p_i, q_i]_{i\in I}, [P_j, Q_j]_{j\notin I})\} \overset{c}{\equiv} \{view_{Miner,\{U_i\}_{i\in I}}([v_i, p_i, q_i]_{i=1}^{n})\} \tag{2.2.3}$$

in which $\overset{c}{\equiv}$ is computational indistinguishability.

This definition means the computation is secure and the honest parties' privacy is guaranteed, if the corrupted participants and the miner learn nothing from the output v and the communication messages of the honest users.

iii. System initialization

Let $(d, \mathbb{E}(\mathbb{Z}_d), \mathbb{O}, G)$ be secure cryptographic parameters.

Each user U_i owns a private value $v_i \in \{0,1\}$. Before starting the PPFC protocol, each party chooses the private keys $p_i, q_i \in [1, d-1]$, and computes the public keys $P_i = p_i G$, $Q_i = q_i G$. Then $P_{,i}, Q_i$ are sent to the miner.

iv. The proposed protocol

The proposed PPFC protocol consists of three main phases described in Protocol 2.6.

Protocol 2.6: A privacy-preserving frequency computation protocol for fully distributed setting

---

Phase 1: Pre-computing

- Miner pre-computes the public values: $P = \sum_{i=1}^{n} P_i$, $Q = \sum_{i=1}^{n} Q_i$
- Miner $\to U_i$: P, Q

Phase 2: Computing the messages

- U_i computes: $M_i = v_i G + q_i P - p_i Q$
- $U_i \to$ Miner: M_i

Phase 3: Secure frequency computation

- Miner computes: $M = \sum_{i=1}^{n} M_i$
- Miner runs Shanks' algorithm to find out v that satisfies $vG = M$

2.2.1.3. Security analysis

i. Proof of Correctness

In this section, it is needed to be shown that the final output of the above PPFC protocol is the sum of all participants' private values. To obtain this, the following theorem is proven.

Theorem 2.2.1. *Protocol 2.6 for privacy-preserving frequency computation exactly counts the number of* $1's$ *values of all parties' inputs.*

Proof. The above theorem means that if the miner finds out a value v, then v is the secure sum of all participants' private values. Indeed, it is supposed that $vG = M$. Then:

$$\begin{aligned}
vG = \; & M \\
= \; & \sum_{i=1}^{n} M_i \\
= \; & \sum_{i=1}^{n} (v_iG + q_iP - p_iQ) \\
= \; & \sum_{i=1}^{n} v_iG + \sum_{i=1}^{n} (q_i \sum_{k=1}^{n} P_k - p_i \sum_{k=1}^{n} Q_k) \\
= \; & \sum_{i=1}^{n} v_iG + \sum_{i=1}^{n} q_i \sum_{k=1}^{n} p_kG - \sum_{i=1}^{n} p_i \sum_{k=1}^{n} q_kG \\
= \; & \sum_{i=1}^{n} v_iG
\end{aligned}$$

Thus, $vG = \sum\limits_{i=1}^{n} v_iG$, and therefore $v = \sum\limits_{i=1}^{n} v_i$. Note that the value of v is not too large, so it can be computed by Shanks' algorithm. □

ii. Privacy Analysis

Firstly, this section proves that the proposed PPFC protocol protects each honest party's privacy in the common semi-honest model. Then, it is shown that this protocol still preserves each honest party's privacy in the case of $(n-2)$ users colluding with the miner.

It is recalled that the point M_i of each user U_i is represented as follows:

$$M_i = (v_iG - p_iQ) + q_i \sum_{i=1}^{n} p_iG \tag{2.2.4}$$

It can be seen that M_i is the first part of an elliptic curve analog of the ElGamal $(P_m + q_iP, q_iG)$ in which $P_m = v_iG - p_iQ$, $\sum p_i$ is the private key, and q_i is uniformly chosen at random from $\{1, ..., d-1\}$. Thus, the new protocol can preserve each honest data user's privacy in the semi-honest model.

Continuously, it is proven that the proposed PPFC protocol protects each party's privacy (even if there are up to $(n-2)$ participants colluding with the miner). The following theorem is stated:

Theorem 2.2.2. *Protocol 2.6 for privacy-preserving frequency computation protects each honest party's privacy against the miner and up to $(n-2)$ corrupted participants.*

Proof. To prove Theorem 2.2.2, it is necessary to be constructed a simulator $\mathbb{M}$ simulating the joint view of the corrupted parties and the miner by a polynomial-time algorithm. Particularly, it is given an algorithm computing the view of the corrupted parties and the miner only using public keys, the result v, corrupted parties' knowledge, and some tuples of ElGamal encryption.

Without loss of generality, it is assumed that U_1, U_2 do not collude and $I = 3,...,n$. In Protocol 2.6, each user only sends a point M_i to the miner, so this algorithm only simulates the computations for M_1, M_2. Below is the computations of $\mathbb{M}$ based on the view of the corrupted parties and the miner using some encryption as its input: $(U_{12}, V_{12}) = (v_2G + q_1(p_2G), p_2.G)$, $(U_{21}, V_{21}) = (v_1G + q_2(p_1G), p_1G)$. Simulator $\mathbb{M}$ computes M_1, M_2 as follows:

$$M_1^{'} = U_{12} + Q_1 \sum_{i \in I} p_i - U_{21} - P_1 \sum_{i \in I} q_i \tag{2.2.5}$$

$$M_2^{'} = U_{21} + Q_2 \sum_{i \in I} p_i - U_{12} - P_2 \sum_{i \in I} q_i \tag{2.2.6}$$

Thus, based on Definition 2.2.1, the proposed `PPFC` protocol is semantically secure. □

2.2.1.4. Performance evaluation

To evaluate the proposed protocol, this section first compares its communication cost and computational complexity with that of the protocol of *Hao et al., 2018* [58] and an elliptic curve analog of the ElGamal system-based variant of the protocol [33]. For convenience, it is noted that three compared protocols execute the Shanks' baby-step giant-step algorithm in the last phase to find out the sum value, and authenticated channels are ready for communicating between the miner/the server and each user.

i. Theoretical evaluation

- *Computational complexity*

Because the computational complexity of the compared protocols mainly relates to their time complexity, this section considers the time complexity required for executing for executing the above protocols. For convenience, the following notations are used:

○ T_M is the time for performing a multiplication operation between a big positive integer and a point on the elliptic curve.

○ T_A is the time for executing an addition operation between two points of the elliptic curve.

○ T_S is the time for executing the Shanks' baby-step giant-step algorithm.

Concerning the variant of Yang et al.'s protocol [33], it takes each user $2T_M$ for pre-computing two public keys. The time for each data user to prepare his/her messages is $2T_M + T_A$. In the pre-computing phase, the time required for the miner is $(2n-2)T_A$. Moreover, the miner must spend $(2n-2)T_A + T_S$ computing the sum value.

In the protocol of *Hao et al., 2018* [58], it takes each user T_M to pre-compute the public value and $T_M + T_A$ to prepare his/her message and the ZKP. Next, the miner spends $(3n-4)T_A$ pre-computing the public keys, and the time for the server to compute the sum value is $(n-1)T_A + T_S$.

Considering the new protocol, in the pre-computation stage each data user consumes $2T_M$ to compute the public value, he continuously spends $2T_M + 2T_A$ preparing his/her message in the first phase. In the second phase, the time for the miner in the pre-computation stage is $(2n-2)T_A$. The proposed protocol's last phase takes the miner $(n-1)T_A + T_S$ to compute the sum value.

The computational complexity comparisons are presented in Table 2.2. It can be seen that each data user in the new solution and the variant of Yang et al.'s proto-

Table 2.2: The computational complexity comparisons among the proposed protocol and the typical protocols.

Protocols	**The time**$^{(1)}$	**The time**$^{(2)}$	**The time**$^{(3)}$	**The time**$^{(4)}$
The variant of Yang et al.'s protocol [33]	$2T_M$	$2T_M + T_A$	$(2n-2)T_A$	$(2n-2)T_A + T_S$
Hao et al., 2018's protocol [58]	T_M	$T_M + T_A$	$3nT_A - 4$	$(n-1)T_A + T_S$
The proposed protocol	$2T_M$	$2T_M + 2T_A$	$(2n-2)T_A$	$(n-1)T_A + T_S$

Note: $^{(1)}$ The time for each user preparing the public keys, $^{(2)}$ The time for each user computing the messages, $^{(3)}$ The time for the miner pre-computing the public keys, $^{(4)}$ The time for the miner computing the frequency value.

col [33] spends the same time $2T_M$ pre-computing the public values, and this time in the protocol of *Hao et al., 2018* is only T_M. However, the time T_M for performing one multiplication operation over the elliptic curve is quite small.

Next, considering the time for each user preparing the messages as shown in the 3^{rd} column of Table 2.2, the time complexity of the proposed protocol is T_A and $(T_A + T_M)$ more than that of the variant of Yang et al.'s protocol [33] and the protocol [58], respectively.

Continuously, as seen in the fourth column of Table 2.2, the variant of Yang et al.'s protocol [33] and the new proposal have the same time for the miner to pre-compute the public keys (i.e. $(2n-2)T_A$). That time is much less than the one in the protocol [58] (i.e. $(3n-4)T_A$).

Lastly, considering the time for the miner/server computing the sum value in the last column of Table 2.2, it takes the new solution and the protocol of *Hao et al., 2018* [58] the same time $(n-1)T_A + T_S$ to the miner to find out the frequency value, and this time of the variant of Yang et al.'s protocol [33] is $(2n-2)T_A + T_S$.

- *Communication cost*

Considering the variant of Yang et al.'s protocol [33], before starting the protocol, each data user needs to send two public keys out the miner. After the miner computes two shared public keys, he/she sends these keys for all data users. In the second phase of the variant of Yang et al.'s protocol [33], each user U_i also needs

Table 2.3: The communication cost comparisons among the typical PPFC protocols.

Protocols	**The number of messages**	**Total size (in bits)**
The variant of Yang et al.'s protocol [33]	$12n$	$12n\|d\|$
The protocol of *Hao et al., 2018* [58]	$6n$	$6n\|d\|$
The new solution	$10n$	$10n\|d\|$

Note: n is the number of users, and $|d|$ is the length of the cryptographic parameter d

to send two values to the miner. Hence, the variant of Yang et al.'s protocol [33] exchanges $12n$ messages.

For the protocol [58], in the pre-computing phase, each user sends one public key to the miner. The miner then sends a specific public key to each user. in the phase 2 of [58], each user submits one message to the miner. Thus, the protocol [58] requires the participants to transfer $6n$ messages.

For the new solution, before it starts, each data user needs to send two public keys (two points) out the miner. The miner then computes two public keys in the first phase, after that he sends them to all users. Continuously, each data user needs to only send a point M_i to the miner in the second phase. As a result, the new solution exchanges $10n$ messages.

Below is Table 2.3 presenting the communication cost comparisons among the compared protocol. It can be seen in this table that the communication cost of the protocol [58] is a half of the variant of Yang et al.'s protocol [33], and it is also less than the new solution's communication cost. However, it is noted that the server of the protocol [58] must send one specific public key to each voter while the miner in the variant of Yang et al.'s protocol [33] and the new solution only sends two shared public keys to all users. Consequently, when working on authenticated channels, the protocol of *Hao et al., 2018* must spend more communication cost (even computational complexity) than the variant of Yang et al.'s protocol [33] and the new proposal.

The next section implements the variant of Yang et al.'s protocol [33], the protocol [58], and the new solution.

ii. Experimental evaluation

- *Experimental setting*

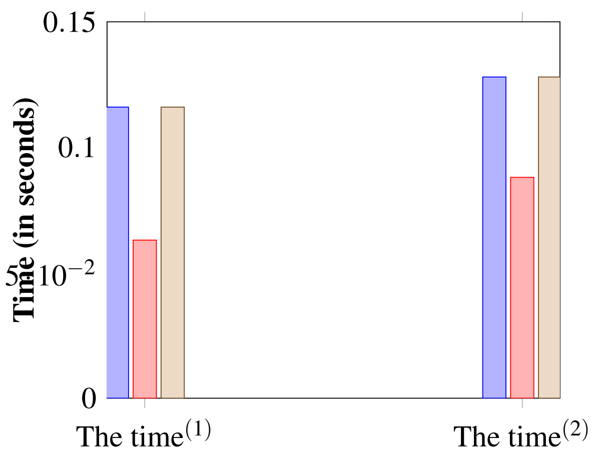
The experiments are implemented on the Lenovo Thinkpad X280 laptop with an Intel core i5 8250U @1.6GHz processor and 8GB memory. In the experiments, all operations are performed over the safe elliptic curve 25519 [74].

It is assumed that all users execute their tasks at the same time, the network latency is not considered in the running time. The experiments are run 50 times with different numbers of users, from 10000 to 50000. Next, the average executing time of the phases in each protocol is calculated using the available library of programming language.

- *Experimental results*

As presented before, in the phase 2 of Yang et al.'s protocol, each user sends two public points to the miner. Based on these values, the miner must additionally perform n addition operations over the elliptic curve. The new proposal is inspirited from the variant of Yang et al.'s protocol [33], but each user U_i computes a unique point M_i in the phase 2 of the new solution, and the miner only computes the sum of all points M_i ($i = \{1,...,n\}$). This improvement only makes each data user's computational complexity negligibly increase, but the miner's computational complexity greatly reduce.

The running time of each user comparisons among three protocols are presented in Figure 2.2. It can be seen that the new proposal and the variant of Yang et al.'s protocol [33] spend the same time preparing the public keys and computing the messages (i.e. about 0.116 seconds and 0.128 seconds, respectively). In addition, those times are larger than the ones in the protocol of *Hao et al., 2018* [58]. The reason of these results is because both the new proposal and the variant of Yang et al.'s protocol [33] require each user to employ two private keys, while that value of the protocol [58] of *Hao et al., 2018* is one. However, the amount of differences are negligible, i.e. 0.053 seconds for preparing the public keys and 0.04 seconds for computing the messages.



🟦 The variant of Yang et al.'s protocol [33] 🟥 Hao et al.'s protocol [58] 🟫 The proposed protocol

$^{(1)}$ The time for preparing the public keys, $^{(2)}$ The time for computing the messages

Figure 2.2: The running time of each user comparisons among the typical `PPFC` protocols.

Continuously, Figure 2.3 describes the time for the miner computing the public keys in three compared protocols. The proposed protocol and the variant of Yang et al.'s protocol [33] also require the same time for the miner computing the public keys. Nevertheless, this time in the protocol [58] of *Hao et al., 2018* is the longest one in the compared protocols, and the amount of difference between the protocol [58] of *Hao et al., 2018* and the new solution is nearly linearly related to the number of data users n, for instance this amount is 1.65 seconds in the case of $n = 10000$ and 7.18 seconds in the case of $n = 50000$. These results are logical to the theoretical comparisons as shown in Table 2.2.

Next, the time for the miner computing frequency value comparisons among three protocols are presented in Figure 2.4. The protocol [58] and the new proposal require the same time for the miner to compute the frequency value. This time is much less than that of the variant of Yang et al.'s protocol [33], and the amount of difference is also nearly linearly related to the number of users n. Particularly, this

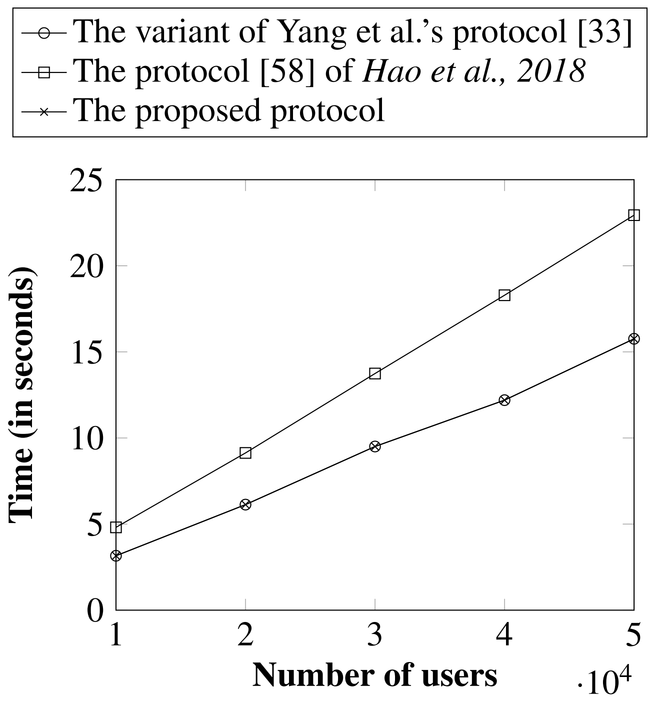


Figure 2.3: The time for the miner/the server computing the public keys comparisons among the typical PPFC protocols.

amount of difference is only about 1.5 seconds in the case of $n = 10000$, but it is up to 7 seconds in the case of $n = 50000$.

In addition, the thesis also compares data volume stored by the miner/server in each protocol presented in Table 2.4 that shows the difference among the compared protocols is negligible. For example, in the case the number of parties is 50000, the miner/server only needs to store 12.2 MB, 6.1 MB, and 9.2 MB in the variant of Yang et al.'s protocol, the protocol of *Hao et al., 2018*, and the new solution, respectively.

According to the above results, it can be stated that the new solution has more advantages than the variant of Yang et al.'s protocol [33], and the protocol [58] of *Hao et al., 2018*. Furthermore, based on the experimental results, it can be stated that the proposed PPFC protocol has a wide applicability.

It needs to be recalled that the above evaluation has not yet considered the cost for establishing authenticated communication between the miner/the server and each user. If this had been done, the communication cost and computational complexity of Hao et al. [58]'s protocol would have significantly increased, because in the pre-

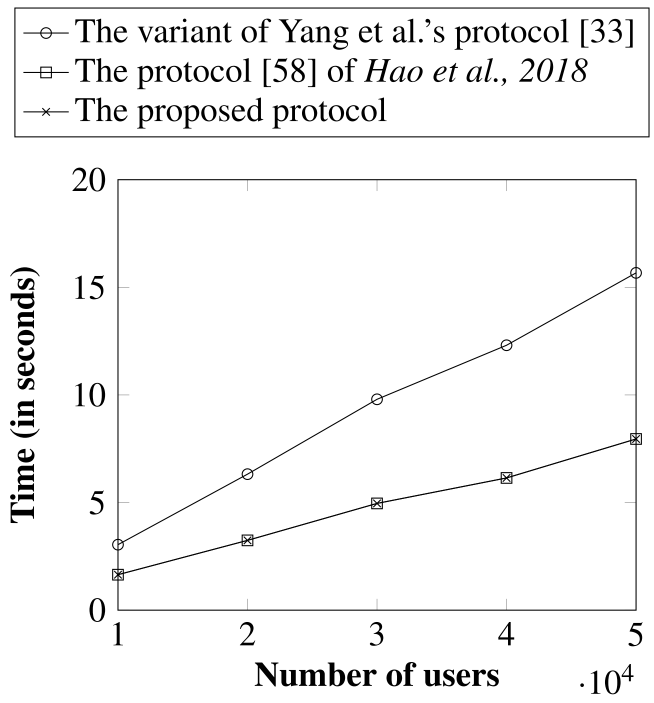


Figure 2.4: The time for the miner/the server computing the frequency value comparisons among the typical PPFC protocols.

computing phase, the server in the protocol [58] of *Hao et al., 2018* must send a specific reconstructed public key to each voter while the miner of the variant of Yang et al.'s protocol [33] and the proposed protocol only needs to sign on the message of two public keys $(P \parallel Q)$ once, then shares $(P \parallel Q)$ and the signature for all without executing any more computational operation.

2.2.2. An efficient approach for secure multi-party sum computation without pre-establishing secure/authenticated channels

In this section, an efficient secure multi-party sum computation protocol without pre-establishing secure/authenticated channels is proposed. This proposal relates to **Publication** 3.

2.2.2.1. Introduction

This contribution of the thesis is to develop a new SMS protocol without pre-establishing secure/authenticated channels that is designed by combining a multi-party sum computation function with a Schnorr signature-derived authentication tech-

Table 2.4: The stored data volume of the miner comparisons among the typical PPFC protocols (in megabytes).

Number of parties / Protocols	10000	20000	30000	40000	50000
The variant of Yang et al.'s protocol [33]	2.4	4.9	7.3	9.8	12.2
The protocol of *Hao et al., 2018* [58]	1.2	2.4	3.7	4.9	6.1
The new solution	1.8	3.7	5.5	7.3	9.2

nique, where these cryptographic tools employ the same private and public keys. Hence, when compared with the existing solutions, the proposed SMS protocol has several following advantages:

- The proposed protocol is highly easy-to-use, because of being directly performed on public networks without pre-installing any cryptographic tool.
- The proposed protocol ensures the result's correctness as well as having high level of security.
- The proposed protocol is applicable to practical problems.

In addition, the proposed protocol requires no communication channel between each tuple of the users. Hence, it is suitable for multi-party distributed models. Furthermore, excepting the key exchange stage, the proposed protocol only requires each data user to send one communication flow to the miner. This advantage makes the new protocol suitable for web applications because the data users only need to submit messages and finish their task.

2.2.2.2. An efficient secure multi-party sum computation protocol without pre-establishing secure/authenticated channels

i. Problem statement

It is recalled that there are a miner and n users $\{U_1, ..., U_n\}$, where each user U_i owns a private value v_i ($i = 1, ..., n$), and the miner wishes to obtain the sum value

$V = \sum_{i=1}^{n} v_i$. For convenience, it is also assumed that $v_i \in \{0,1\}$ ($\forall i \in [1,n]$), because if the inputs are common integers then the proposed protocol still efficiently calculates the sum value, and in the case that the private inputs are quite large integers, the data scaling process should be executed before performing the new SMS protocol. For an instance, instead of calculating the sum of four persons' income $\{10000\$, 20000\$, 35000\$, 50000\$\}$, it may be computed the sum of values $\{10, 20, 35, 50\}$.

ii. Definition of security

The new SMS protocol is based on the common semi-honest model [3], so the security definition is stated as follows:

Definition 2.2.2. *Assume that each data user U_i has the private keys (x_i, y_i) and the corresponding public keys (X_i, Y_i). A protocol protects each data user's privacy against t corrupted parties and the miner in the common semi-honest model if, for all $I \subseteq \{1,...,n\}$ such that $||I|| = t$, there exists a probabilistic polynomial-time algorithm $\mathbb{M}$ such that*

$$\{\mathbb{M}(V, [v_i, x_i, y_i]_{i \in I}, [X_j, Y_j]_{j \notin I})\} \overset{c}{\equiv} \{view_{miner, \{U_i\}_{i \in I}}([v_i, x_i, y_i]_{i=1}^{n})\} \tag{2.2.7}$$

in which $\overset{c}{\equiv}$ is computational indistinguishability.

To reach the mentioned objectives, this section first develops a multi-party sum computation function that is re-designed from the original protocol in [33]. Next, instead of using popular cryptographic techniques such as digital signature standard (DSS), RSA signature scheme, this section designs a Schnorr signature-based authentication method (after here called Schnorr signature-based ZKP) employing the same parameters with the above multi-party sum computation function. By combining this authentication method with the multi-party sum computation function, it can be obtained a novel SMS protocol that can securely protect the data users' privacy and to guarantee the output's correctness without pre-establishing any authenticated channel. Additionally, this way helps to optimize the proposed protocol's performance. The next section presents the main stages of the new protocol.

iii. System initialization

The new protocol uses the following parameters:

• Let $(\mathbb{G}, p, q, g)$ be public parameters as mentioned in Section 1.1.3.

• Each data user U_i has already held a private key x_i and the corresponding public key is $X_i = g^{x_i}$ in his/her digital certificate issued by a trusted certification authority. Each data user randomly chooses a secret number $y_i \in \{1, ..., q-1\}$ and computes $Y_i = g^{y_i}$. Note that y_i is only used once and X_i is also sent to the miner before the new protocol starts.

• $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ that is a secure hash function [75].

Before the protocol starts, each data user U_i directly sends Y_i to the miner without his/her (ZKP), because of the existence of the user authentication phase (see in Protocol 2.7).

Next, the miner pre-computes:

$$X = \prod_{i=1}^{n} X_i \, ; Y = \prod_{i=1}^{n} Y_i \tag{2.2.8}$$

After that, the miner shares $M = (X \parallel Y)$ and the Schnorr signature of M for all data users via public networks. This work ensures that no one can forge him, but only takes the miner a negligible cost.

iv. The proposed protocol

The proposed protocol contains three main phases described as follows.

• *Data submission phase*

In this phase, each participant U_i first authenticates the miner's Schnorr signature, then he encrypts his/her secret value v_i by computing $P_i = \frac{g^{v_i} X^{y_i}}{Y^{x_i}}$. Next, he computes his/her Schnorr signature-based ZKP $(r_i = Y_i; s_i \equiv y_i - x_i H(r_i \parallel P_i) \pmod{q})$ and only sends P_i, s_i to the miner, because Y_i has been sent before starting the new protocol.

- *User authentication phase*

In the phase 2, to authenticate each data user U_i, firstly, the miner computes $r_i^{'} = g^{s_i} X_i^{\gamma_i}$, where $\gamma_i = H(r_i \parallel P_i)$. If $r_i = r_i^{'}$, the miner accepts the user U_i, and vice versa.

- *Secure n-parties sum computation phase*

The miner aggregates $K = \prod_{i=1}^{n} P_i$. Next, he performs Shanks' algorithm to achieve the sum value V that satisfies $g^V = K$.

Three main phases of the new `SMS` protocol are presented in Protocol 2.7.

Protocol 2.7: A secure n-parties sum protocol without pre-establishing secure/authenticated channels.

Phase 1: Data submission

- Each user verifies the miner's Schnorr signature on $M = (X \parallel Y)$
- Each user U_i computes: $P_i = \frac{g^{v_i} X^{y_i}}{Y^{x_i}}, r_i = Y_i, s_i \equiv y_i - x_i H(r_i \parallel P_i) \pmod{q}$
- Each user $U_i \rightarrow$ Miner: P_i, s_i

Phase 2: User authentication

- Miner computes: $\gamma_i = H(r_i \parallel P_i), r_i^{'} = g^{s_i} X_i^{\gamma_i}$
- Miner authenticates each user U_i by verifying the equation: $r_i \overset{?}{=} r_i^{'}$

Phase 3: Secure n-parties sum computation

- Miner computes: $K = \prod_{i=1}^{n} P_i$
- Miner executes **Shanks' algorithm** to find out V that satisfies $g^V = K$

2.2.2.3. Security analysis

As mentioned before, the new protocol is composed from two sub-protocols employing the same cryptography parameters, in which the first is the multi-party sum computation function that is re-designed from the original protocol in [33], and the second is the Schnorr signature-based authentication tool. According to the composition theorem mentioned in Chapter 1, it needs to be shown that both sub-protocols are secure.

i. The sub-protocol for computing the sum value

- *Proof of correctness*

This section proves that the final output of the sub-protocol for computing the sum value is the sum of all participants' private values, if they follow the protocol's rules. Hence, the following theorem is stated.

Theorem 2.2.3. *The sub-protocol for computing the sum value exactly computes the sum of all data users' private values.*

Proof. It needs to be showed that if the miner finds out the value V satisfying the equation $g^S = K$, then V is the sum of all data users' private values.

Assume that $g^V = K$. Then:

$$\begin{aligned} g^V &= K \\ &= \prod_{i=1}^{n} P_i \\ &= \prod_{i=1}^{n} \frac{g^{v_i} X^{y_i}}{Y^{x_i}} \\ &= g^{\sum_{i=1}^{n} v_i} \prod_{i=1}^{n} \frac{X^{y_i}}{Y^{x_i}} \\ &= g^{\sum_{i=1}^{n} v_i} \prod_{i=1}^{n} \frac{(\prod_{j=1}^{n} X_j)^{y_i}}{(\prod_{j=1}^{n} Y_j)^{x_i}} \\ &= g^{\sum_{i=1}^{n} v_i} \prod_{i=1}^{n} \frac{(g^{\sum_{j=1}^{n} x_j})^{y_i}}{(g^{\sum_{j=1}^{n} y_j})^{x_i}} \\ &= g^{\sum_{i=1}^{n} v_i} \frac{g^{\sum_{j=1}^{n} x_j \sum_{i=1}^{n} y_i}}{g^{\sum_{j=1}^{n} y_j \sum_{i=1}^{n} x_i}} \\ &= g^{\sum_{i=1}^{n} v_i} \end{aligned}$$

Hence, $g^V = g^{\sum_{i=1}^{n} v_i}$, and thus $V = \sum_{i=1}^{n} v_i$. □

Therefore, the output result is the sum of all participants' private values.

- *Privacy analysis*

In the following section, it is proven that the sub-protocol for computing the sum value securely protects each honest data user's privacy in the semi-honest model. Considering this sub-protocol, each message P_i of the user U_i is represented by the following equation:

$$P_i = (g^{v_i} Y^{-x_i}) X^{y_i} \tag{2.2.9}$$

It can be seen that the message P_i in Equation (2.2.9) is the first part of an ElGamal encryption (mX^{y_i}, g^{y_i}), correspondingly $m = g^{v_i} Y^{-x_i}$, $\sum x_i$ is the private key and y_i is uniformly chosen. Thus, the sub-protocol for computing the sum value securely preserves each honest data user's privacy in the common semi-honest model.

Continuously, it is shown that the sub-protocol for computing the sum value still securely preserves each honest participant's privacy in the case of $(n-2)$ parties colluding with the miner. Hence, the following theorem is stated.

Theorem 2.2.4. *The sub-protocol for computing the sum value protects each honest data user's privacy against the miner and up to $(n-2)$ corrupted data users.*

Proof. As mentioned before, it needs to be given an algorithm computing the joint view of the corrupted data users and the miner using only public keys, the output result V, corrupted data users' knowledge, and some ElGamal encryptions.

Without loss of generality, it is assumed that U_1 and U_2 do not collude. In the sub-protocol for computing the sum value, each user only shares two values (P_i, s_i) for the miner where the value s_i is a random value, since the private keys x_i, y_i are random. As a result, the algorithm only needs to simulate the computation for P_1 and P_2. The algorithm computing the view of the miner and the corrupted data users using several ElGamal encryptions $(u_{12}, v_{12}) = (g^{v_1} g^{x_2 y_1}, g^{x_2}), (u_{21}, v_{21}) = (g^{v_2} g^{x_1 y_2}, g^{x_1})$ as its input. $\mathbb{M}$ computes P_1 and P_2 is presented as follows:

$$P_1' = \frac{u_{12} Y_1^{\sum_{i \in I} x_i} g^{V - \sum_{i \in I} v_i}}{u_{21} X_1^{\sum_{i \in I} y_i}} \tag{2.2.10}$$

$$P_2^{'} = \frac{u_{21} Y_2^{\sum_{i \in I} x_i} g^{V - \sum_{i \in I} v_i}}{u_{12} X_2^{\sum_{i \in I} y_i}} \tag{2.2.11}$$

Thus, according to Definition 2.2.2, the sub-protocol for computing the sum value is semantically secure. $\square$

ii. The sub-protocol for authenticating each user

Regarding the sub-protocol for authenticating each user U_i using the tuple $\{P_i, r_i = g^{y_i}, s_i \equiv y_i - x_i H(r_i \parallel P_i) \pmod{q}\}$, it needs to be shown that (i) each user U_i with the tuple $\{P_i, r_i, s_i\}$ is properly authenticated by proving Theorem 2.2.5, and (ii) the sub-protocol for authenticating each user is secure against possible attacks in the random oracle model.

- *Proof of correctness*

Theorem 2.2.5. *The miner of the protocol presented in Protocol 2.7 properly authenticates each user.*

Proof. To show Theorem 2.2.5's correctness, the following equation is considered.

$$g^{s_i}.X_i^{\gamma_i} \equiv r_i \pmod{p} \; . \tag{2.2.12}$$

where $\gamma_i = H(r_i \parallel P_i)$, $r_i = g^{y_i}$, and $s_i \equiv y_i - x_i H(r_i \parallel P_i) \pmod{q}$.

Indeed,

$$\begin{aligned} g^{s_i} X_i^{\gamma_i} &\equiv g^{y_i - x_i H(r_i \| P_i)} X_i^{H(r_i \| P_i)} && \pmod{p} \\ &\equiv g^{y_i - x_i H(r_i \| P_i)} g^{x_i H(r_i \| P_i)} && \pmod{p} \\ &\equiv g^{y_i} && \pmod{p} \\ &\equiv r_i && \pmod{p} \end{aligned}$$

Thus, Theorem 2.2.5 is proven. $\square$

Next, based on the security standard for digital signature [41, 76], this section proves the security of the sub-protocol for authenticating each user by showing that this authentication protocol is secure against the adaptively chosen-message attack.

It is recalled that the sub-protocol for authentication purpose is directly derived from Schnorr signature scheme [42]. In more detail, the elements $(x_i, y_i, P_i, r_i, s_i)$ in the sub-protocol presented in Protocol 2.7 are perfectly equal to Schnorr signature scheme's elements (s, r, m, x, y) presented in Figure 2 of the work [42]. In the other words, Schnorr signature scheme and the sub-protocol for authenticating each user have the same level of security.

In the other hand, Pointcheval and Stern pointed in [77, 78] that if an existential forgery of the Schnorr signature scheme, under an adaptively chosen-message attack in the random oracle model, has non-negligible probability of success, then the discrete logarithm in subgroups can be solved in polynomial time. Meanwhile, the cryptographic parameters (p, q, g) are chosen to ensure that solving discrete logarithm problems in subgroups are hard (see in Section 1.1. This means Schnorr signature scheme is secure against the adaptively chosen-message attack. Thus, it can be stated that the sub-protocol for authenticating each user is also secure against the adaptively chosen-message attack.

2.2.2.4. Performance evaluation

Recall that in the new protocol, each data user U_i only computes a unique value P_i related to v_i, and the public values $Y_i = g^{y_i}$, P_i are used in both the user authentication phase and secure multi-party sum computation stage. Additionally, the proposed protocol does not use any more the private key but (x_i, y_i). Hence, the communication cost and computational complexity of the new protocol reduce significantly.

This section compares the performance of the new solution with the typical protocols of Yang et al. in [33] and of *Hao et al., 2014* in [13], briefly named Yang's protocol and Hao2014's protocol, respectively. They are chosen for the comparisons, because they have the highest level of privacy among the existing `SMS` protocols.

Firstly, the computational complexity and communication cost of the above protocols are evaluated. Secondly, the experiments are deployed with the best implementation, after that the running time of the miner and each user in three protocols is measured. Finally, the experimental results are analyzed.

To guarantee the fairness for the comparisons, it is supposed the following settings:

- Yang's and Hao2014's protocols use the Schnorr signature scheme [42] to ensure that the messages transferred between the miner and each user are authenticated.
- The miner and each data user of the compared protocols have already owned their private and public keys for creating and verifying their Schnorr signatures/ZKPs, and these public keys have been exchanged before the protocols start.
- RIPEMD-160 hash function [79] is used in the experiments.
- For each communication flow in three protocols, there is a unique Schnorr signature/ZKP (if existing), and it is the Schnorr signature/ZKP of data concatenated from all communication messages in that flow.
- The compared protocols use Shanks' algorithm to obtain the sum value from the discrete logarithm problem.

i. Theoretical evaluation

In this section, the thesis considers a comparative study on the compared SMS protocols in terms of communication cost and computational complexity.

- *Computational complexity*

This section considers the times T_e, T_m, T_i for executing modular exponentiation, modular multiplicative inverse, and modular multiplication operations (respectively) with big integers. The time complexity T_h of the hash function H is also listed. T_e is the most expensive operation among these ones. Moreover, the time complexity for executing Shanks' algorithm is also considered, and this time is denoted as T_S.

Concerning Yang's protocol, it takes each data user $2T_e$ to pre-compute two public values and $(T_e + T_m + T_h)$ to compute his/her Schnorr signature on these public values in the pre-computation phase. Next, the miner spends $(2T_e + T_m + T_h)$ authenticating each user and $2(n-1)T_m$ computing two shared public values for all users. Simultaneously, this entity spends $(T_e + T_m + T_h)$ creating the Schnorr signature. Continuously, each user must consume $(5T_e + 3T_m + 2T_h)$ to authenticate the miner and prepare m_i, h_i with his Schnorr signature. In the user authentication stage, the time for the miner verifying each user is $(2T_e + T_m + T_h)$. Lastly, the miner spends $((2n-1)T_m + nT_i + T_S)$ calculating the sum value.

In Hao2014's protocol, it takes each user T_e for pre-computing his/her public value and $(T_e + T_m + T_h)$ to compute his/her Schnorr signature. Continuously, it takes the miner $(2T_e + T_m + T_h)$ to authenticate each user and $((3n-4)T_m + T_i)$ to compute the public values. Then it is required the miner $n(T_e + T_m + T_h)$ to sign on n public values. Next, each user spends $(4T_e + 3T_m + 2T_h)$ authenticating the miner's message and preparing his/her message with the ZKP. To authenticate each user and compute the sum value, the miner must consume $(2T_e + T_m + T_h)$ and $((n-1)T_m + T_S)$, respectively.

Considering the proposed protocol, each user U_i only spends T_e computing the public value $Y_i = g^{y_i}$ in the pre-computation stage. Next, it takes the miner $(T_e + (2n-1)T_m + T_h)$ to compute the shared public values X, Y and the Schnorr signature of $M = (X \parallel Y)$. Continuously, each user must consume $(4T_e + 4T_m + 2T_h)$ to authenticate the miner's message M and compute his/her messages in the data submission phase. Then it takes the miner $(2T_e + T_m + T_h)$ to authenticate each user and $((n-1)T_m + T_S)$ to compute the sum value.

The comparisons of each user's computational complexity are presented in Table 2.5. It can be seen in the pre-computation phase that the time complexity of each user in the proposed protocol is the smallest among the compared protocols. This is understandable, because each user U_i only sends the public value $Y_i = g^{y_i}$ without attaching his/her Schnorr signature-based to the miner.

Continuously, regarding the time for each data user for authenticating the miner and preparing the messages as shown in the column 3 of Table 2.5, that time of

Table 2.5: The comparisons of each user's computational complexity among the proposed protocol and the typical protocols.

Protocols	**The time**$^{(1)}$	**The time**$^{(2)}$
Yang's protocol [33]	$3T_e + T_m + T_h$	$5T_e + 3T_m + 2T_h$
Hao2014's protocol [13]	$2T_e + T_m + T_h$	$4T_e + 3T_m + 2T_h$
The proposed protocol	T_e	$4T_e + 4T_m + 2T_h$

Note: $^{(1)}$ The time for each user in the pre-computation phase, $^{(2)}$ The time for each user in the data submission phase.

the new solution is T_e and $T_e + T_m$ more than that of Hao2014's protocol and Yang's protocol, respectively. However, these differences are negligible.

Table 2.6: The miner's computational complexity comparisons among the proposed protocol and the typical protocols.

Protocols	**The time**$^{(1)}$	**The time**$^{(2)}$	**The time**$^{(3)}$
Yang's protocol [33]	$(2n+1)T_e + (3n-1)T_m + (n+1)T_h$	$n(2T_e + T_m + T_h)$	$(2n-1)T_m + nT_i + T_S$
Hao2014's protocol [13]	$3nT_e + (5n-4)T_m + T_i + 2nT_h$	$n(2T_e + T_m + T_h)$	$(n-1)T_m + T_S$
The proposed protocol	$T_e + (2n-1)T_m + T_h$	$n(2T_e + T_m + T_h)$	$(n-1)T_m + T_S$

Note: $^{(1)}$ The time for the miner in the pre-computation phase, $^{(2)}$ The time for the miner in the user authentication phase, $^{(3)}$ The time for the miner in the secure n-parties sum computation phase.

Next, the miner's computational complexity comparisons are described in Table 2.6. The new solution and Yang's protocol require the miner to respectively consume $T_e + (2n-1)T_m + T_h$ and $(2n+1)T_e + (3n-1)T_m + (n+1)T_h$ in the pre-computation phase, while that time of Hao2014's protocol is up to $3nT_e + (5n-4)T_m + T_i + 2nT_h$. There exists this difference, since the miner must compute the specific public key and the Schnorr signature for each user in Hao2014's protocol. Consequently, the time for the miner in the pre-computation phase of Hao2014's protocol is lengthy, if the number of user n is large.

Lastly, it can be seen in the third columns of Table 2.6 that it takes all protocols the same time (i.e. $n(2T_e + T_m + T_h)$) for the miner in the user authentication phase. However, the fourth column of Table 2.6 shows that Hao2014's protocol and the new proposal require the same time (i.e. $(n-1)T_m + T_S$) for the miner in the last phase while that time of Yang's protocol [33] is up to $(2n-1)T_m + nT_i + T_S$.

- *Communication cost*

This section compares the communication cost among the new proposal and the typical protocols in [13,33]. In this section, some notations $|p|$, $|q|$ and $|h|$ are the large prime p's length, the length of the small prime q, and the length of the output of the hash function H, respectively.

Regarding the communication messages in Yang's protocol, before starting the protocol, each data user U_i sends $\{X_i, Y_i\}$ and his/her Schnorr signature to the miner, then he/she receives the public values $\{X, Y\}$ and a Schnorr signature on $(X \parallel Y)$ from the miner. In the data submission phase, each data user U_i sends $\{m_i, h_i\}$ to the miner. Nevertheless, to be correctly authenticated, each user U_i needs to attach a Schnorr signature on $(m_i \parallel h_i)$ together with $\{m_i, h_i\}$. Thus, each user's communication cost is $(4|p| + 2|q| + 2|h|)$ bits, and the miner's communication cost is $n(2|p| + |q| + |h|)$ bits.

In Hao2014's protocol [13], each data user first sends a public value and a Schnorr signature to the server. Then the miner sends a specific public key attached with a Schnorr signature to each user. Next, he/she sends out the ciphertext of his/her private value and a Schnorr signature to the server. Hence, each user's communication cost is $(2|p| + 2|q| + 2|h|)$ bits, and the miner's communication cost is $n(|p| + |q| + |h|)$ bits.

Considering the new solution, before starting the protocol, each data user U_i only sends Y_i to the miner. Next, he receives the shared public values $\{X, Y\}$ and a Schnorr signature on $(X \parallel Y)$. Continuously, each data user U_i submits two messages $\{P_i, s_i\}$ to the miner. Thus, each user's communication cost is $(2|p| + |q|)$ bits, and the miner's communication cost is $n(2|p| + |q| + |h|)$ bits.

The comparisons of each data user's communication cost are shown in Ta-

Table 2.7: The comparisons of each user's communication cost among the proposed protocol and the typical protocols.

Protocols	**The number of flows of comm.**	**Total size (in bits)**
Yang's protocol	2	$4\lvert p\rvert + 2\lvert q\rvert + 2\lvert h\rvert$
Hao2014's protocol	2	$2\lvert p\rvert + 2\lvert q\rvert + 2\lvert h\rvert$
The proposed protocol	2	$2\lvert p\rvert + \lvert q\rvert$

Table 2.8: The comparisons of the miner's communication cost among the proposed protocol and the typical protocols.

Protocols	**The number of flows of comm.**	**Total size (in bits)**
Yang's protocol	n	$n(2\lvert p\rvert + \lvert q\rvert + \lvert h\rvert)$
Hao2014's protocol	n	$n(\lvert p\rvert + \lvert q\rvert + \lvert h\rvert)$
The proposed protocol	n	$n(2\lvert p\rvert + \lvert q\rvert + \lvert h\rvert)$

ble 2.7. It is clear that the compared protocols require each data user to transfer the same number of communication flows, where amount of data (in bits) required for each data user in the new protocol is the lowest in the compared protocols.

Table 2.8 presents the miner's communication cost in the compared protocols. The miner of all compared protocols sends the same number of communication flows. However, the amount of data (in bits) transferred by the miner in Hao2014's protocol is $n\lvert p\rvert$ bits less than Yang's protocol and the new proposal.

ii. Experimental evaluation

- *Experimental setting*

The experiments are run on the Lenovo Thinkpad X280 laptop. In the experiments, the prime numbers p, q are $2048, 256$ bits length, respectively, as recommended in the document [80].

The experiments are run 50 times for different numbers of data users from 200000 to 1000000. Next, the average running time of four stages for each protocol is calculated using the available library of programming language.

- *Experimental results*

Figure 2.5 presents the time for each data user in the pre-computation and data submission phases. Particularly, the experimental results in Figure 2.5 show that each user in the new solution only spends about 20 milliseconds pre-computing the parameters, while this time in Yang's protocol and Hao2014's protocol is almost 60 and 40 milliseconds, respectively. Continuously, the time for each user to prepare his/her messages is almost 80 milliseconds in Hao2014's protocol and the new solution. This time is about 100 milliseconds in Yang's protocol. These results are logical to the theoretical comparisons as shown in Table 2.5

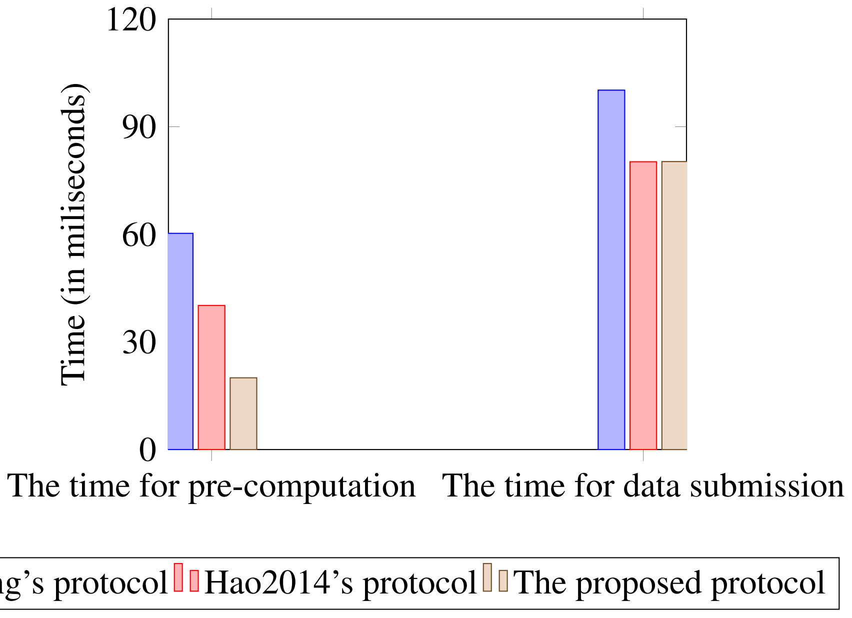


Figure 2.5: The running time of each user comparisons among the proposed protocol and the typical protocols.

Next, the running time of the miner is presented in Figures 2.6, 2.7, 2.8.

In particular, Figure 2.6 shows the time for the miner in the pre-computation phase. That time in Yang's and Hao2014's protocols is extremely much larger than the one in the new proposal. For example, in the case $n = 1000000$, the proposed protocol only takes the miner about 100 seconds in the pre-computation phase. However, Yang's and Hao2014's protocols take the miner up to 40190 and 60330 seconds, respectively. The reason for this difference is that the miner in Yang's and Hao2014's protocols must verify the message containing each user's public values. Especially, Hao2014's protocol requires the miner to sign on each user's specific public value to send to each user.

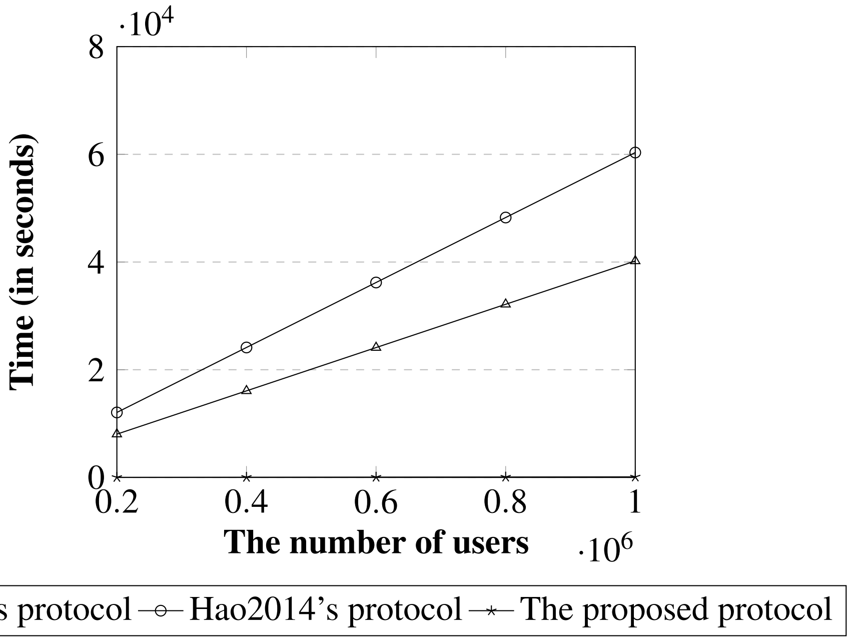


Figure 2.6: The time of the pre-computation phase comparisons among the proposed protocol and the typical protocols.

Continuously, the time for the miner in the user authentication phase of the compared protocols is presented in Figure 2.7. Because each user in the proposed protocol uses the Schnorr signature-based authentication method, it can be easily seen that all of three protocols require the miner the same running time in the user authentication phase, and that time is linearly related to the number of users n. This is also logical to the theoretical comparisons as shown in Table 2.5.

Lastly, Figure 2.8 presents the time for the secure n-parties sum phase. As seen in this figure, it can be determined that Hao2014's protocol and the new proposal spend the same time for the last phase, while Yang's protocol requires the miner the

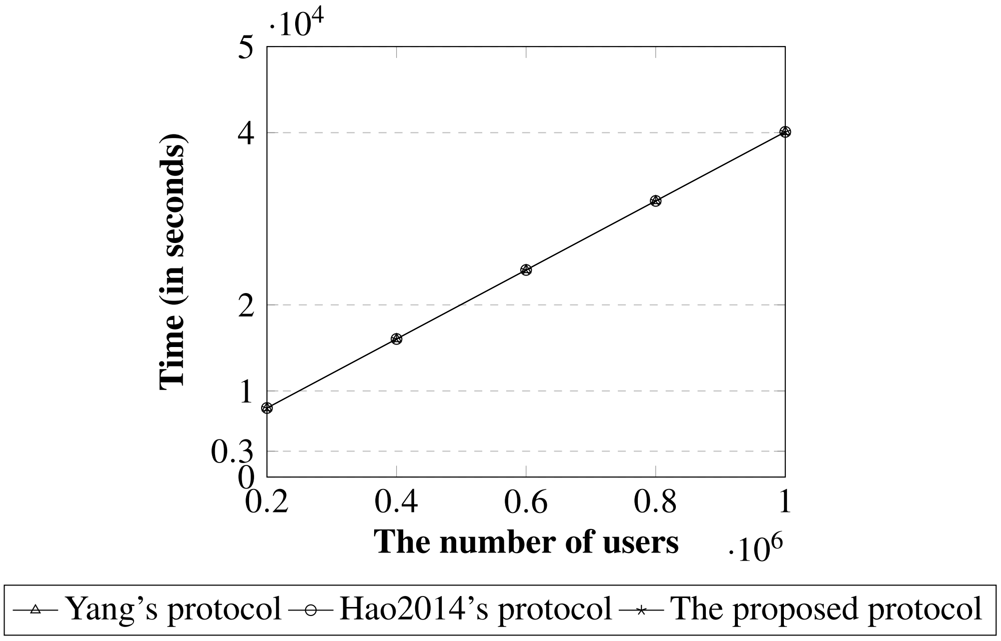


Figure 2.7: The time of the user authentication phase comparisons among the proposed protocol and the typical protocols.

longest time for the secure n-parties sum phases in the compared protocols. This can be explained that in the last phase of Yang's protocol, the miner must compute ${h_i}^{-1}$ and $m_i{h_i}^{-1}$ to obtain $\frac{m_i}{h_i}$ with $i = \{1, 2, ..., n\}$ while the miner in the other does not need to do this task.

In addition, the thesis also compares data volume stored by the miner/server in each protocol presented in Table 2.9 showing that in the cases of number of parties, that data volume of the proposed protocol is only a half of Yang's and Hao2014's protocols. For example, in the case the number of parties is 1000000, the miner/server of the new protocol only needs to use 537.9 MB, while that amount of Yang's and Hao2014's protocols is up to about 1075.7 MB.

In summary, based on the above experimental results, it can be stated that the new SMS protocol has more advantages than the others. Thus, when compared to the typical SMS protocols, the new proposal is the most suitable solution for applications in practice.

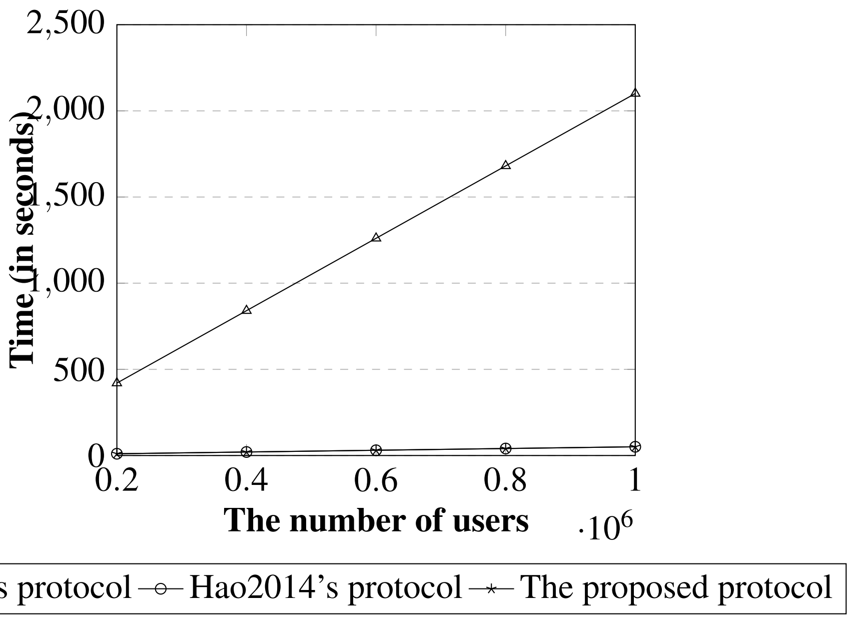


Figure 2.8: The time of the secure n-parties sum phase comparisons among the proposed protocol and the typical protocols.

Table 2.9: The stored data volume of the miner comparisons among the proposed protocol and the typical protocols (in megabytes).

Number of parties / Protocols	200000	400000	600000	800000	1000000
Yang's protocol [33]	215.1	430.3	645.4	860.6	1075.7
Hao2014's protocol [13]	215.1	430.3	645.4	860.6	1075.7
The proposed protocol	107.6	215.1	322.7	430.3	537.9

2.2.3. Secure multi-sum computation protocol

This section presents an efficient secure multi-sum computation protocol that can securely compute multiple sum values only in a unique round of computation. This proposal of the thesis is related to **Publication** 4.

2.2.3.1. Introduction

As mentioned above, the aim of the secure multi-sum computation or the secure agregation problem is to obtain multiple sum values in one round of computation. Instead of performing secure sum/privacy-preserving frequency computation protocols multiple times or using expensive homomorphic encryption systems (e.g. the Paillier encryption, the LWE-based encryption schemes), this section provides a new secure building block called the secure multi-sum computation protocol that has capacity to simultaneously and efficiently compute multiple sum values.

The main idea of the new secure multi-sum computation protocol is to flexibly and delicately combine the secure sum protocols in one round of computation. In particular, before starting the proposed protocol, the tuples of private and public keys have been prepared. After that, each data provider encrypts his/her private values into ciphertexts to submit them to the miner once. This is detailed in the following sections.

2.2.3.2. The secure protocol for multi-sum computation

i. Problem statement

This section propounds a novel secure multi-sum computation protocol. Here, the computational model consists of np parties $\{U_1,...,U_{np}\}$ where each party U_i has a set V_i of ns private small/medium non-negative integers $\{v_i^1,...,v_i^{ns}\}$, and a miner needs to achieve the set SUM consisting of ns sum values $\{Sum_1 = \sum_{i=1}^{np} v_i^1,...,Sum_{ns} = \sum_{i=1}^{np} v_i^{ns}\}$. Instead of performing privacy-preserving frequency computation or secure sum protocols ns times, the new protocol is designed as follows:

- Each participant prepares the set Prv_i of nk private keys $\{Prv_i^1,...,Prv_i^{nk}\}$ and the Pub_i set of nk corresponding public keys $\{Pub_i^1,...,Pub_i^{nk}\}$ at the beginning (nk is computed from Equation 2.2.14) and uses these keys to encrypt his/her set V_i of ns input values into ns ciphertexts. Each party then only needs to submit these ciphertexts to the miner one time.

- The miner aggregates ns ciphertexts of sum values from all messages received from the participants, then extracts ns sum values. To perform this task, the miner needs to solve ns discrete logarithm problems by only executing the brute-force algorithm once (as described in Appendix B), since these problems have the same space of solutions (i.e. $\{0,1,...,np\}$).

ii. Definition of security

The new secure multi-sum computation protocol is based on the semi-honest model [3], so the security definition is stated as follows:

Definition 2.2.3. *A secure multi-sum computation protocol protects each user's privacy against t corrupted parties and the miner in the semi-honest model if, for $\forall I \subseteq \{1,...,np\}$ such that $\|I\| = t$, there exists a probabilistic polynomial-time algorithm $\mathbb{M}$ such that*

$$\{\mathbb{M}(SUM,[V_i,Prv_i]_{i\in I},[Pub_j]_{j\notin I})\} \overset{c}{\equiv} \{view_{miner,\{U_i\}_{i\in I}}([V_i,Prv_i]_{i=1}^{n})\} \tag{2.2.13}$$

where $\overset{c}{\equiv}$ is computational indistinguishability.

In the following section, the proposed secure multi-sum computation protocol is presented in detail.

iii. System initialization

Let $(\mathbb{G},p,q,g)$ be cryptographic parameters as mentioned in Section 1.1.3. These parameters are known by every participant.

iv. The proposed protocol

The main stages of the new secure multi-sum computation protocol are presented in Protocol 2.8.

Protocol 2.8: A secure protocol for computing multi-sum in one round of computation

```
1. Phase 1: The users $U_i$ do
forall $i$ where $1 \le i \le np$ do
    forall $j$ where $1 \le j \le nk$ do
        $Prv_i^j = Random(1, q-1)$
        $Pub_i^j = g^{Prv_i^j}$
        Sends to Miner: $Pub_i^j$
    end
end

2. Phase 2: Miner does
forall $j$ where $1 \le j \le nk$ do
    $Pub^j = 1$
    forall $i$ where $1 \le i \le np$ do
        $Pub^j = Pub^j * Pub_i^j$
    end
    Sends to all $U_i$: $Pub^j$
end

3. Phase 3: The users $U_i$ do
forall $i$ where $1 \le i \le np$ do
    $j = 1$
    forall $t$ where $1 \le t \le nk - 1$ do
        forall $k$ where $t + 1 \le k \le nk$ do
            $p_i^j = g^{v_i^j} (Pub^t)^{Prv_i^k} (Pub^k)^{q - Prv_i^t}$
            if $j == ns$ then
                break
            else
                $j{+}{+}$
            end
        end
    end
    Sends to Miner: $p_i^j$
end

4. Phase 4: Miner does
forall $j$ where $1 \le j \le ns$ do
    $K^j = 1$
    forall $i$ where $1 \le i \le np$ do
        $K^j = K^j * p_i^j$
    end
end
Solves the problems $g^{Sum_j} = K^j$ ($j \in \{1, ..., ns\}$) by running the brute-force algorithm once
```

Protocol 2.8 shows that each private value v_i^s of U_i is encrypted by two tuples of private and public keys. As a result, if each party uses nk tuples of keys, he/she can encrypt up to $\mathbb{C}_2^{nk}$ private values. Hence, each party only needs to prepare nk tuples of private and public keys, where *nk* is calculated by the following equation:

$$\boxed{nk = \left\lceil \frac{1}{2} + \sqrt{2ns + \frac{1}{4}} \right\rceil} \tag{2.2.14}$$

Clearly, the value *nk* is often much smaller than *ns*.

2.2.3.3. Security analysis

i. Proof of Correctness

To prove the proposed protocol's correctness, the thesis states and proves Theorem 2.2.6.

Theorem 2.2.6. *Protocol 2.8 correctly computes multiple sum values only in one round of computation.*

Proof. Indeed, it is supposed that $g^{Sum_j} = K^j$. Then:

$$\begin{aligned}
g^{Sum_j} &= \prod_{i=1}^{np} p_i^j \\
&= \prod_{i=1}^{np} g^{v_i^j} (Pub^t)^{Prv_i^k} (Pub^k)^{q-Prv_i^t} \\
&= \prod_{i=1}^{np} \frac{g^{v_i^j} (Pub^t)^{Prv_i^k}}{(Pub^k)^{Prv_i^t}} \\
&= \prod_{i=1}^{np} \frac{g^{v_i^j} (\prod_{i=1}^{np} Pub_i^t)^{Prv_i^k}}{(\prod_{i=1}^{np} Pub_i^k)^{Prv_i^t}} \\
&= \prod_{i=1}^{np} \frac{g^{v_i^j} (\prod_{i=1}^{np} g^{Prv_i^t})^{Prv_i^k}}{(\prod_{i=1}^{np} g^{Prv_i^k})^{Prv_i^t}} \\
&= \prod_{i=1}^{np} \frac{g^{v_i^j} (g^{\sum_{i=1}^{np} Prv_i^t})^{Prv_i^k}}{(g^{\sum_{i=1}^{np} Prv_i^k})^{Prv_i^t}} \\
&= \frac{g^{\sum_{i=1}^{np} v_i^j} (g^{\sum_{i=1}^{np} Prv_i^t})^{\sum_{j=1}^{np} Prv_j^k}}{(g^{\sum_{i=1}^{np} Prv_i^k})^{\sum_{j=1}^{np} Prv_j^t}} \\
&= g^{\sum_{i=1}^{np} v_i^j}
\end{aligned}$$

Thus, $Sum_j = \sum_{i=1}^{np} v_i^j$. $\square$

ii. Privacy analysis

In this section, it is proved that the proposed protocol securely protects honest data parties' privacy in the common semi-honest model.

Recall that, each data user U_i only sends the values p_i^j ($j \in \{1,...,ns\}$) in one communication flow where each value p_i^j is represented by the following equation:

$$p_i^j = (g^{v_i^j}(Pub^k)^{q-Prv_i^t})(Pub^t)^{Prv_i^k} \tag{2.2.15}$$

It can be seen that the value p_i^j in Equation 2.2.15 is the first part of an ElGamal encryption $(m(Pub^t)^{Prv_i^k}, g^{Prv_i^k})$ with $m = g^{v_i^j}(Pub^k)^{q-Prv_i^t}$, the private key is $\sum Prv_i^t$ and Prv_i^k is randomly chosen in $\{1,...,q-1\}$. Hence, the proposed protocol can securely protect honest parties' privacy in the common semi-honest model.

Next, it is shown that the new proposal still securely preserves the honest parties' privacy in the case of $(np-2)$ parties controlled by the miner. The thesis states the following theorem:

Theorem 2.2.7. *Protocol 2.8 securely protects each data user's privacy against up to $(np-2)$ corrupted parties and the miner.*

Proof. Without the loss of generality, it can be assumed that U_1 and U_2 are honest. Hence, the algorithm $\mathbb{M}$ only needs to simulate the computation for the values p_1^j and p_2^j ($j \in \{1,...,ns\}$) via the following equations:

$$P_1^j = \frac{u_{12}(Pub_1^k)^{\sum_{i \in I} Prv_i^t} g^{Sum_j - \sum_{i \in I} v_i^j}}{u_{21}(Pub_1^t)^{\sum_{i \in I} Prv_i^k}} \tag{2.2.16}$$

$$P_2^j = \frac{u_{21}(Pub_2^k)^{\sum_{i \in I} Prv_i^t} g^{Sum_j - \sum_{i \in I} v_i^j}}{u_{12}(Pub_2^t)^{\sum_{i \in I} Prv_i^k}} \tag{2.2.17}$$

in which $(u_{12}, v_{12}) = (g^{v_1^j} g^{Prv_2^t Prv_1^k}, g^{Prv_2^t})$, $(u_{21}, v_{21}) = (g^{v_2^j} g^{Prv_1^t Prv_2^k}, g^{Prv_1^t})$

Thus, according to Definition 2.2.3 and the compositiontheorem, the proposed protocol is semantically secure. □

2.2.3.4. Performance evaluation

This section evaluates the new proposal and the solutions created by executing multiple times the protocols of Yang et al. in [33], Hao et al., 2014 [13], and Hien et al. in the third publication (denoted as Yang's-based solution, Hao2014's-based solution, and Hien's-based solution, respectively). The above three solutions are chosen for the comparisons, since they have the same high level of security. For each compared solution, the communication cost and computational complexity of four phases are considered. It is recalled that the parameters np, ns are the number of users and the number of sum values, the number of tuples of private & public keys nk in each solution is calculated from ns.

At the first step, the compared solutions are theoretically evaluated (i.e. computational complexity and communication cost aspects). Next, this section measures and compares the running time of these solutions.

i. Theoretical evaluation

• *Computational complexity*

For convenience, it is denoted T_e, T_m, T_i as the time for performing a modular exponentiation operation, executing a modular multiplication, and computing a modular multiplicative inverse, respectively, where the most expensive operation is T_e. Additionally, the time T_S for running Shanks' algorithm is considered in Hao2014's-based and Hien's-based solutions.

For Yang's-based solution, it takes each data holder $2ns.T_e$ to prepare the necessary keys in the phase 1 and $ns(2T_e + T_m)$ for encrypting his/her array of ns private values in the phase 3. The miner in Yang's-based solution spends $2ns(np-1)T_m$ for computing the shared public keys in the phase 2 and $ns(npT_i + (2np + ns - 2)T_m)$ for finding the sum values in the phase 4.

Concerning Hao2014's-based solution, the computational costs for each data user preparing the necessary keys in the phase 1 and encrypting his/her array of ns private values in the phase 3 are nsT_e and $ns(T_e + T_m)$, respectively. The miner of Hao2014's-based solution spends $ns((3np-4)T_m + T_i)$ computing the public keys in the phase 2 and $ns((np-1)T_m + T_S)$ finding the sum values in the phase 4.

For Hien's-based solution, each data holder spends nkT_e preparing the necessary keys in the phase 1 and $ns(2T_e + 2T_m)$ encrypting his/her array of ns private values in the phase 3. The miner consumes $nk(np-1)T_m$ for computing the shared public keys in the phase 2 and $ns((np-1)T_m + T_S)$ for extracting the sum values in the phase 4.

Considering the new solution, the computational costs of each data user preparing the necessary keys in the phase 1 and encrypting his/her private values in the phase 3 are nkT_e and $ns(2T_e + 2T_m)$, respectively. The miner spends $nk(np-1)T_m$ computing the shared public keys in the phase 2 and $(ns(np-1) + ns - 1)T_m$ extracting the sum values in the phase 4.

The computational complexity comparisons are presented in Table 2.10. Hien's-based solution and the new proposal take the same time nkT_e for each data user computing the necessary keys in the phase 1 and the same time $nk(np-1)T_m$ for the miner computing the public keys in the phase 2, while these times in Hao2014's-based solution are nsT_e and $ns((3np-4)T_m + T_i)$, and the ones in Yang's-based solution are $2nsT_e$ and $2ns(np-1)T_m$. Among the compared solutions, the time complexity for each data user encrypting ns private values in the third phase of Hao2014's-based solution is smallest. It is easy to understand the above differences, since Yang's and Hao2014's-based solutions must use $2ns$ and ns tuples of keys, respectively, for jointly computing ns the sum values, while the new proposal and Hien's-based solution only need to employ nk tuples of private and public keys. Additional, among the compared solutions, the computational complexity for the miner extracting the sum values in the proposed solution is smallest.

• *Communication cost*

Table 2.10: The computational complexity comparisons among the new proposal and the typical solutions.

Solutions	**The time for the phase** 1	**The time for the phase** 2	**The time for the phase** 3	**The time for the phase** 4
Yang's-based solution	$2nsT_e$	$2ns(np-1)T_m$	$ns(2T_e+T_m)$	$ns(npT_i+(2np+ns-2)T_m)$
Hao2014's-based solution	nsT_e	$ns((3np-4)T_m+T_i)$	$ns(T_e+T_m)$	$ns((np-1)T_m+T_S)$
Hien's-based solution	nkT_e	$nk(np-1)T_m$	$ns(2T_e+2T_m)$	$ns((np-1)T_m+T_S)$
The new solution	nkT_e	$nk(np-1)T_m$	$ns(2T_e+2T_m)$	$(ns(np-1)+ns-1)T_m$

This section compares the communication cost among the compared solutions. It is noted that $|p|$ is denoted as the length of public keys.

Regarding the communication messages in Yang's-based solution, each data user sends his/her $2ns$ public keys in the phase 1 and ns tuples of ciphertexts in the phase 3 to the miner. The miner sends $2ns$ shared public keys to all data users in the phase 2. These messages are $|p|$ bits length, so the communication cost of Yang's-based solution is $6np.ns\,|p|$ bits.

Considering the communication cost of Hao2014's-based solution, each data user sends ns public keys in the 1^{st} phase and ns ciphertexts in the 3^{rd} phase to the miner. The miner of Hao2014's-based solution also sends ns specific public keys to each data holder in the phase 2. Thus, the communication cost of Hao2014's-based solution is $3np.ns\,|p|$.

For both the new proposal and Hien's-based solution, each data holder sends nk public keys in the 1^{st} phase and ns ciphertexts in the 3^{rd} phase to the miner. The miner of the above solutions sends nk shared public keys to all data holders in the phase 2. Thus, the new proposal and Hien's-based solution have the same communication cost $np\{2nk+ns\}\,|p|$ bits.

The communication cost comparisons are shown in Table 2.11 that the compared solutions send the same number of flows of communication. Moreover, the amount of transferred data (in bits) of Hien's-based solution and the new proposal is $np\{2nk+ns\}\,|p|$, while the one of Yang's and Hao2014's-based solutions is $6np*ns\,|p|$ and $3np*ns\,|p|$, respectively. Obviously, if ns is large, then the communication cost of Yang's and Hao2014's-based solutions is much larger than that of Hien's-based and the new ones.

Table 2.11: The communication cost comparison among the new proposal and the typical solutions.

Solutions	**The number of comm. flows**	**Total size (in bits)**
Yang's-based solution	$3np$	$6np * ns\,\lvert p\rvert$
Hao2014's-based solution	$3np$	$3np * ns\,\lvert p\rvert$
Hien's-based solution	$3np$	$np\{2(\left\lceil \frac{1}{2}+\sqrt{2ns+\frac{1}{4}} \right\rceil)+ns\}\,\lvert p\rvert$
The new solution	$3np$	$np\{2(\left\lceil \frac{1}{2}+\sqrt{2ns+\frac{1}{4}} \right\rceil)+ns\}\,\lvert p\rvert$

ii. Experimental evaluation

• *Experimental setting*

In the experiments, all compared solutions are implemented in the Python language of Anaconda environment on the Lenovo Thinkpad X280 laptop. The prime numbers p, q are $2048, 256$ bits length, respectively. For convenience, it is assumed that each private input value is 1, and the space of discrete logarithm problems' solutions is from 0 to np.

The experiments are run 50 times for different tuples (number of users, number of sums), i.e. (1000, 500), (2000, 1000), (3000, 1500), (4000, 2000), (5000, 2500). Next, the average running time of four phases for each solution is calculated using the available library of Python language.

• *Experimental results*

Firstly, Figure 2.9 shows the number of private keys which each user of the compared solutions uses in the different cases of number of users and number of sums. The new proposal and Hien's-based solution employ the same small number of private keys while these numbers in Hao2014's-based and Yang's-based solutions are very large. For example, in the case of number of users = 5000 and number of sums = 2500, the new proposal and Hien's-based solution only uses 72 private keys, but Hao2014's-based and Yang's-based solutions need to employs up to 2500 and 5000 private keys, respectively. This is because both the new proposal and Hien's-based solution can re-use private keys without security concerns. As a result, the new solution not only has the low time for preparing the necessary keys, but also saves the cost of keys management.

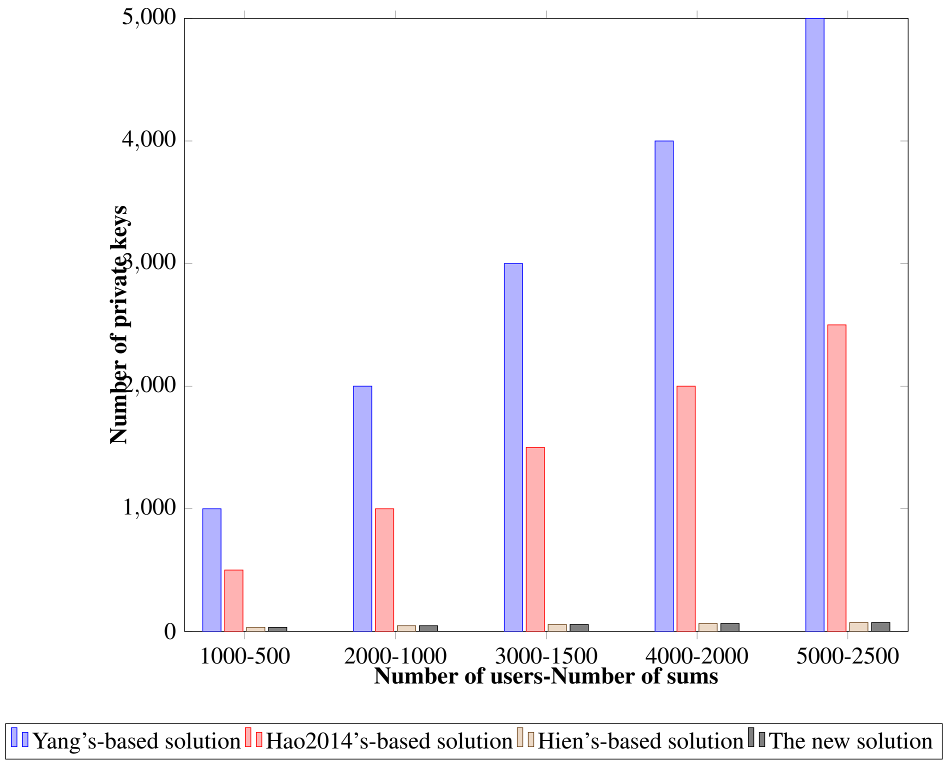


Figure 2.9: The number of private keys comparisons among the compared solutions.

Continuously, Figure 2.10 presents the total running time of each user in the compared solutions. It is easy to see that the total running time of each user in Yang's-based solution is the largest when compared with others. The reason of this is because each user of Yang's-based solution must use two private keys to encrypt a private value, and these two keys cannot be used. Figure 2.10 also shows that Hien's-based solution and the new proposal require the same total running time for each user, and that time is only slight more than that of Hao2014's-based solution. For example, in the case of number of users = 5000 and number of sums = 2500, this difference is only about 1 second.

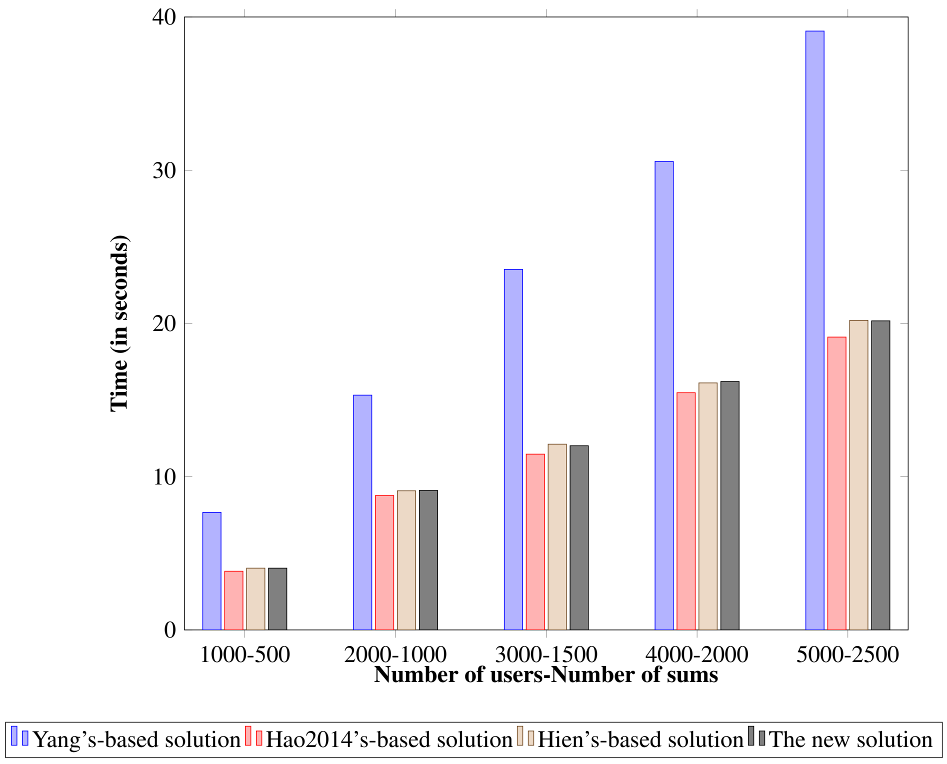


Figure 2.10: The total running time of each user comparisons among the compared solutions.

Next, the comparisons on the running time for the miner to compute the public keys are illustrated in Figure 2.11. It can be seen that the new proposal and Hien's-based solution spend the small time for the miner to compute the public keys while that time of Hao2014's-based and Yang's-based solutions is extremely large. For example, in the case of number of users = 5000 and number of sums = 2500, it only takes the new proposal and Hien's-based solution about 5 seconds, but Yang's-based and Hao2014's-based solutions correspondingly require the miner up to 343 and 571 seconds to compute the public keys. This is because the number of private keys used in the new proposal and Hien's-based solution is much smaller than that of Yang's-based and Hao2014's-based solutions.

Next, the running time for the miner to compute the sum values comparisons among the compared solutions are described in Table 2.12. It is clear that the new so-

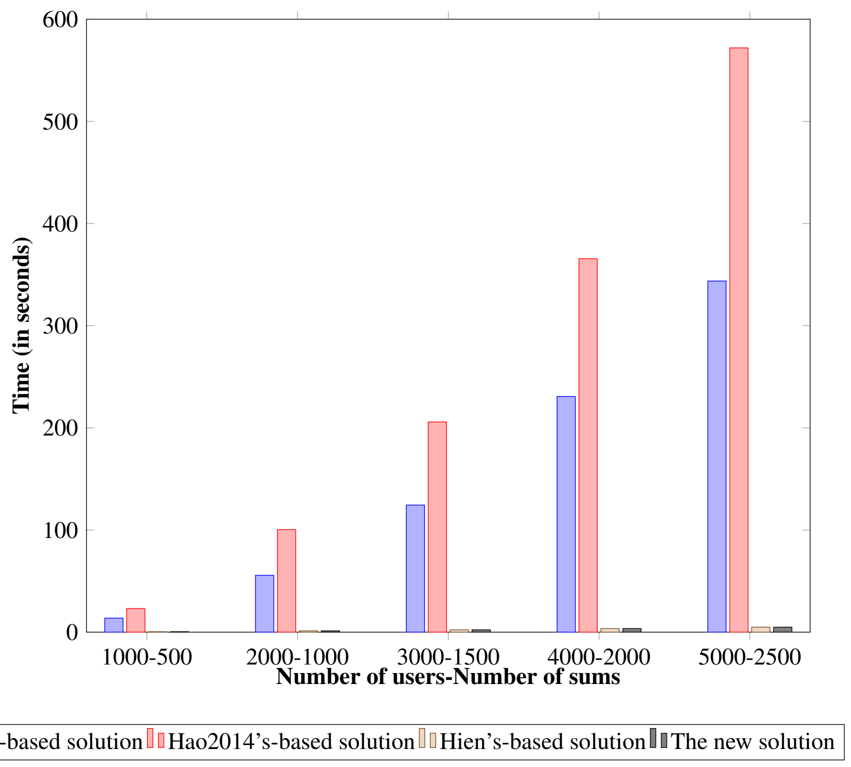


Figure 2.11: The running time for the miner to compute the public keys comparisons among the compared solutions.

lution requires the smallest running time for the miner computing the sum values (see the bold values in Table 2.12), and that time is about 3 and 23 seconds less than that in Hao2014's-based and Hien's-based solutions in the cases that (number of users, number of sums) is (1000, 500) and (5000, 2500), respectively. Especially, Yang-based solution requires the extremely large time for the miner to compute the sum values in the last phase (see the underlined values in Table 2.12). This is understandable, since the miner of Yang-based solution must perform the slow brute force algorithm *ns* times to find out *ns* sum values.

Lastly, the thesis also compares data volume stored by the miner/server in each solution presented in Table 2.13 showing that in the cases of number of parties and sum values, the data volume of Hien's-based solution and the new one is smallest among the compared solutions. For example, in the case the number of parties is 5000 and the number of sum values is 2500, the miner/server of the new protocol only needs to use 3139.7 MB, while that amount of Yang's-based and Hao2014's-based solutions is up to about 12209.5 MB and 9155.3 MB, respectively.

Table 2.12: The running time for the miner to compute the sum values comparisons among the compared solutions (in seconds).

Number of users-Number of sums **Solutions**	**1000-500**	**2000-1000**	**3000-1500**	**4000-2000**	**5000-2500**
Yang's-based solution	823.142	3217.324	7299.532	13320.492	19310.021
Hao2014's-based solution	10.000	35.556	74.314	128.160	195.650
Hien's-based solution	10.005	35.004	74.102	128.533	195.647
The new solution	**7.104**	**27.588**	**62.085**	**110.262**	**172.094**

Table 2.13: The stored data volume of the miner comparisons among the compared solutions (in megabytes).

Number of users-Number of sums **Solutions**	**1000-500**	**2000-1000**	**3000-1500**	**4000-2000**	**5000-2500**
Yang's-based solution	488.8	1954.1	4396.0	7814.5	12209.5
Hao2014's-based solution	366.2	1464.8	3295.9	5859.4	9155.3
Hien's-based solution	130.1	510.8	1139.7	2015.6	3139.7
The new solution	130.1	510.8	1139.7	2015.6	3139.7

In summary, considering all above experimental results, it can be stated that the proposed solution has more advantages than the others. Thus, when compared to the typical solution, the new proposal is the most suitable one for applications in practice.

2.3. Conclusion

This chapter has provided a background of secure multi-party computation field and comprehensively analyzed the most typical existing work related to this study. Based on the analysis results, three new protocols have been propounded for the privacy-preserving frequency computation, secure multi-party sum computation without pre-establishing secure/authenticated channels, and secure multi-sum computation problems, respectively. It has been proven that the proposed protocols are secure in the semi-honest model. The evaluations also show their efficiency. Thus, the proposed protocols have the capability to be applied in practical problems requiring to securely compute frequency or sum values, such as secure electronic voting scheme, privacy-preserving data mining/machine learning techniques (e.g. K-means clustering, Apriori association rule mining, Naive Bayes classification, ID3 decision tree) for distributed data models.

CHAPTER 3. DEVELOPING NEW SOLUTIONS BASED ON SECURE MULTI-PARTY SUM COMPUTATION PROTOCOLS FOR PRACTICAL PROBLEMS

Based on the proposed protocols presented in Chapter 2, this chapter constructs solutions for two very practical problems that are the secure end-to-end decentralized voting scheme and the privacy-preserving Naive Bayes classification technique in the horizontally distributed data setting. The proposed solutions are not only considered the aspect of security but also evaluated their performance. These proposals of the thesis are related to **Publications** 2, 4, and 5.

3.1. An efficient solution for the secure electronic voting scheme without pre-establishing authenticated channel

In this section, an efficient solution for electronic voting scheme without pre-establishing authenticated channel is proposed. This proposal has related to **Publication** 2.

3.1.1. Introduction

Generally, the electronic voting scheme (also called e-voting system) is one of the most classical cryptographic applications. Such scheme uses electronic systems for casting and counting votes [81]. Currently, there have been two approaches to construct an electronic voting scheme [81]: (1) based on homomorphic cryptosystems (e.g. [12, 13, 58, 81, 82]), and (2) based on anonymous channels (e.g. [83, 84]).

According to methods for voting, e-voting systems are separated into two types [81, 85]: centralized elections in which the voters employ voting machines at polling stations to submit their ballots, and decentralized elections (internet voting or remote voting) where the voters send their ballots to the voting server through a network.

In this section, the thesis focuses on the end-to-end (E2E) decentralized electronic voting system for the small or medium scale elections that have been investigated in the previous work [12, 58, 81, 82]. There are four key properties for such voting systems [85–89]:

- **Accuracy**: all valid votes are listed correctly, and the e-voting protocol's output is the correct result. Nobody can remove, alter, or duplicate the voters' ballots.
- **Secrecy/Privacy**: Nobody knows each voter's choice beyond himself/herself (even in the case of some voters colluding with the voting server).
- **Verifiability**: Anyone can verify/prove the voting result's correctness.
- **Efficiency**: each voter uses the lowest cost executing his/her tasks, and the performance of the electronic voting system is good enough to be implemented in practice.

3.1.2. Related work

Generally, most of existing solutions for the E2E decentralized electronic voting scheme are based on cryptographic protocols. However, these solutions have a lot of drawbacks as analyzed in the following section.

The first E2E decentralized electronic voting scheme has been introduced by Kiayias and Yung in [81]. This protocol is self-tallying, dispute-free, and supports perfect ballot secrecy. Nevertheless, its computational cost for each voter is heavy. After that, the solution [82] was propounded with the aim of improving computational complexity. In the protocol [82], the computational cost of the i^{th} voter depends on that of the $(i-1)^{th}$ voter, hence the executing time of [82] greatly increases.

To reduce the number of computational rounds, the authors proposed [12] based on the two rounds anonymous veto protocol [12]. However, each voter's computational complexity in [12] linearly increases with the number of voters. Furthermore, the communication cost of this protocol is bounded by $O(n^2)$. After that, the DRE-based e-voting [13] was proposed to guarantee the integrity property. In the protocol [13], the voting server computes the restructured public values P_i for all voters to reduce each voter's cost. Nevertheless, the total time of the voting task is not improved.

One of the most practical e-voting solution named Helios [90] that can tally integrity. This solution has a number of drawbacks as analyzed in [13] such as a vulnerable Java plug-in, each voter's expensive computation cost, low level of privacy, or much weakness.

Inspired from the work [59], the authors of [58] propounded the verifiable classroom voting system. Although, the protocol [58]'s communication cost is lower than that of [12, 13], but the execution time of this protocol is still equivalent to that of [13].

In briefly, the performance of the existing electronic voting solutions is quite poor. Furthermore, to prevent an adversary from modifying the ballots, most of existing decentralized electronic voting schemes [12, 13, 58, 81, 82] use authenticated channels for the voters sending their ballots to the voting server. This is inconvenient, since these solutions cannot be directly implemented on public networks. In addition, this makes the performance of electronic voting schemes significantly reduce.

The contribution of this section is to develop an efficient and secure solution for the E2E decentralized e-voting system. The new scheme is also based on cryptography, but because of improvements, this solution has the following advantages:

- No trusted party involves in the e-voting system. The semi-honest voting server is only used for computing the public parameters for the voters.
- Each voter clicks his/her choice (e.g. 'yes/no' buttons) to broadcast his/her encrypted ballot to the voting server through public networks (e.g. Internet) without pre-establishing any authenticated channel. No one knows his/her selection beyond himself/herself (even if there are up to some voters colluding with the voting server).
- The new solution is efficient and convenient, because each voter only interacts with the voting server once excepting the pre-processing stage.

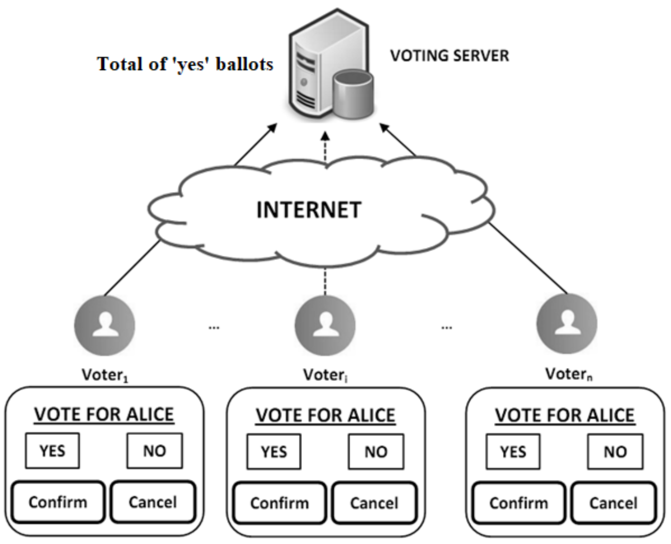


Figure 3.1: The single-candidate E2E decentralized electronic voting model.

3.1.3. Preliminaries

3.1.3.1. System model

In the model as illustrated in Figure 3.1, there are n voters $\{U_1,...,U_n\}$, in which each voter U_i $(i = 1,...,n)$ clicks 'yes/no' buttons on the voting website to vote his/her 'yes/no' ballot v_i ($1/0$, respectively). The voting server works as a bulletin board enabling any body to read the data viewed on it. In the addition, the voting server also consists of high performance computers. The system model assumes no communication among the voters, and each voter only uses a public network for interacting with the voting server.

As mentioned before, our system is secure if the following requirements are satisfied:

- The output of the electronic voting protocol is the correct voting result.
- Nobody knows each voter's ballot (1 or 0?) beyond himself/herself (even if the voting server colludes with several voters).

3.1.3.2. Definition of security

In the proposed voting system model, each voter is supposed as a semi-honest participant. Thus, the following security definition that is similar to the ones in [1,3, 33] is stated:

Definition 3.1.1. *Assume that each voter U_i has two private keys p_i, q_i and the corresponding public keys P_i, Q_i. An electronic voting protocol protects each voter's privacy against t corrupted voters and the voting server in the common semi-honest model if, $\forall I \subseteq \{1,...,n\}$ such that $\|I\| = t$, there exists a probabilistic polynomial-time algorithm $\mathbb{M}$ such that*

$$\{\mathbb{M}(v, [v_i, p_i, q_i]_{i\in I}, [P_j, Q_j]_{j\notin I})\} \overset{c}{\equiv} \{view_{Server,\{U_i\}_{i\in I}}([v_i, p_i, q_i]_{i=1}^{n})\}.$$

where $\overset{c}{\equiv}$ is computational indistinguishability, and v_i is the voter U_i's ballot.

3.1.4. A secure end-to-end electronic voting scheme

In the new e-voting system, each voter needs to know that his/her ballot submitted to the server by himself. This is the most important requirement, since it ensures that the voting result is correct. In the proposed electronic voting system, an authentication method is redesigned from the work [42]. Combining this method with the privacy-preserving frequency computation protocol presented in Section 2.3.1, an efficient solution for the decentralized e-voting scheme is created.

In the proposed solution, it is assumed that all participants agree to employ the following parameters. Let $(q, E(\mathbb{Z}_q), \mathbb{O}, G)$ be secure cryptographic parameters.

The new electronic voting scheme contains the main stages as follows:

- *Preparation phase*: each voter has already owned a private key $p_i \in [1, q-1]$ and the public key point $P_i = p_iG$. On the other hand, the voting server has kept a tuple of private and public keys to sign and verify its signature. Before the electronic voting protocol starts, each voter randomly chooses a private value $q_i \in [1, q-1]$, and he/she computes the corresponding public point $Q_i = q_iG$. The information of Q_i is also sent to the voting server before starting the protocol. The voting server pre-computes the shared public values $P = \sum_{i=1}^{n} P_i$, $Q = \sum_{i=1}^{n} Q_i$ and sends them to every voter.

- *Ballot submission phase*: each voter computes his/her encrypted ballot with its ZKP.
- *Voter authentication phase*: the voting server verifies each voter's ZKP.
- *Vote counting phase*: the voting server computes the number of *'yes'* votes for the candidate based on the voters' encrypted ballots.

In this proposal, H is a secure hash function as mentioned in [75].

It is also necessary to be noted that this section denotes x_P as the value corresponding to the point P of the curve $E(\mathbb{Z}_q)$, and vice verse (using a method of imbedding plaintexts mentioned in [43]).

The main phases of the single-candidate decentralized e-voting system based on privacy-preserving frequency computation protocol are presented in Protocol 3.1.

Protocol 3.1: A single-candidate decentralized electronic voting scheme based on privacy-preserving frequency computation protocol

Phase 0: Preparation

- Each voter U_i chooses $p_i, q_i \in \{1, ..., q-1\}$
- Each voter U_i computes: $P_i = p_i G$, $Q_i = q_i G$
- $U_i \rightarrow$ Voting server: P_i, Q_i
- Voting server pre-computes the shared public values: $P = \sum_{i=1}^{n} P_i$, $Q = \sum_{i=1}^{n} Q_i$
- Voting server $\rightarrow$ All voters: P, Q

Phase 1: Ballot submission

- Each voter U_i computes: $M_i = v_i G + q_i P - p_i Q$, $r_i = x_{Q_i}$, $s_i = q_i - p_i H(x_{M_i} \parallel r_i)$
- $U_i \rightarrow$ Voting server: M_i, s_i

Phase 2: Voter authentication

- Voting server publishes M_i, s_i on its bulletin board
- Voting server authenticates U_i by:
  - Computing $\gamma_i = H(x_{M_i} \parallel r_i)$, $R_i^{'} = s_i G + \gamma_i P_i$
  - Verifying the equation $r_i \overset{?}{=} x_{R_i^{'}}$

Phase 3: Vote counting

- Voting server computes: $M = \sum_{i=1}^{n} M_i$
- Executes Shanks' algorithm to obtain the output v satisfying $vG = M$

3.1.5. Security analysis

In this section, the most important security properties of the proposed electronic voting scheme (i.e. accuracy, privacy, and verifiability) are considered.

3.1.5.1. Accuracy

In this section, it needs to be proved that the output v of the proposed electronic voting scheme in Protocol 3.1 is the total 'yes' votes. Because the proposed electronic voting system is based on the privacy-preserving frequency computation protocol presented in Section 2.3.1, this section only proves that each voter and his/her ballot is properly verified, as well as considering possible attacks where the adversary tries to create the fake voting result by deleting, duplicating, or modifying the voters' ballots.

i. Proof of the authentication method's correctness

In the new voting scheme, it needs to be proved that the authentication method modified from the work [42] is correct.

Indeed, the equation $R_i^{'} = s_iG + \gamma_iP_i$ is transformed as follows:

$$\begin{aligned} R_i^{'} &= s_iG + \gamma_iP_i \\ &= s_iG + H(x_{M_i} \parallel r_i)P_i \\ &= (q_i - p_iH(x_{M_i} \parallel r_i))G + H(x_{M_i} \parallel r_i)p_iG \\ &= q_iG \\ &= Q_i \end{aligned}$$

Hence, $x_{R_i^{'}} = x_{Q_i}$, and thus the equation $r_i = x_{R_i^{'}}$ is true and the authentication method modified from the work [42] is correct.

Next, it is shown that if he tries to delete, modify, or duplicate the voters' ballots, then the adversary must face the hard problems.

ii. Ballots modification attacks

The attacks using this way are classified into two types: forging ballots and ZKP, and recovering the private keys attacks.

- *Recovering the private keys attacks:* an adversary tries to analyze the equation $Q_i = s_iG + \gamma_iP_i$ to obtain the private keys (p_i, q_i) of the voter U_i, then he generates another valid ballot of the voter U_i (even he can learn U_i's selection). This task is equivalent to solving a discrete logarithm problem.
- *Forging ballots and ZKP attacks:* assume that a forger tries to create the

values $\{H(x_{M_i'} \parallel x_{Q_i}); Q_i'; s_i'\}$ satisfying the system of equations:

$$\begin{cases} \gamma_i' & = H(x_{M_i'} \parallel x_{Q_i}) \\ Q_i' & = s_i'G + \gamma_i' P_i \end{cases} \tag{3.1.1}$$

For example, the forger may randomly choose k and t in $\{1, ..., q-1\}$, he then computes: $Q_i' = kG + tP_i; s_i' = k; \gamma_i' = t$.

Nevertheless, to cheat the voting server, the adversary must compute M_i' from the equation $\gamma_i' = H(x_{M_i'} \parallel x_{Q_i})$, he then sends the set (M_i', s_i') to the voting server. This is hard, because of the secure hash function H.

In another way, the forger tries to find $M_i' \neq M_i$ such that $H(x_{M_i'} \parallel x_{Q_i}) = H(x_{M_i} \parallel x_{Q_i})$, he then sends the values (P_i', s_i) to cheat the voting server. However, this is hard because of the secure hash function H.

iii. Ballots deletion attacks

These attacks cannot be performed, because the bulletin board helps any voter to easily detect his/her ballot's absence.

iv. Ballots duplication attacks

Similarly, the adversary cannot execute these attacks, because of using the bulletin board.

In briefly, the proposed electronic voting scheme ensures the correctness of the voting result. No one can duplicate, delete, modify the voters' ballots.

3.1.5.2. Privacy

As mentioned before, the proposed electronic voting scheme is based on the privacy-preserving frequency computation protocol presented in Section 2.3.1. Thus, the proposed electronic voting scheme is also semantically secure.

3.1.5.3. Verifiability

In the scheme presented in Protocol 3.1, the voting server shows the encrypted ballots M_i of all voters on the bulletin board. Anybody can easily verify the correctness of voting result by performing the phase 3 of the new proposal (see in Protocol 3.1). Thus, the proposed scheme satisfies the verifiability requirement.

3.1.6. Experimental evaluation

This section considers the running time between the proposed solution and one of the most typical electronic voting schemes [58] (Hao's scheme, for short). It is assumed that all voters interact with the voting server at the same time, and the network latency is ignored.

3.1.6.1. Experimental setting

The experiments are implemented on the Lenovo Thinkpad X280 laptop. In the experiments, all public key operations are defined over the curve 25519 [74], and the secure hash function $RIPEMD-160$ is chosen.

To measure the running time of the compared solutions, the experiments are run 50 times with different numbers of voters, from 2000 to 10000. Next, the average executing time of four phases (i.e. preparation, ballot submission, voter authentication, and vote counting) for each solution is calculated using the available library of programming language.

3.1.6.2. Experimental results

The experimental results are detailed in Figures 3.2 and 3.3.

Figure 3.2 presents the total running time of each voter in Hao's scheme and the proposed solution. Particularly, the proposed scheme requires each voter to spend almost 170 milliseconds performing his/her tasks, while that time in Hao's solution is about 204 milliseconds. The reason of this difference is because each voter in Hao's solution must send its signature on the public point in the preparation phase.

Of courses, the difference here is not too large.

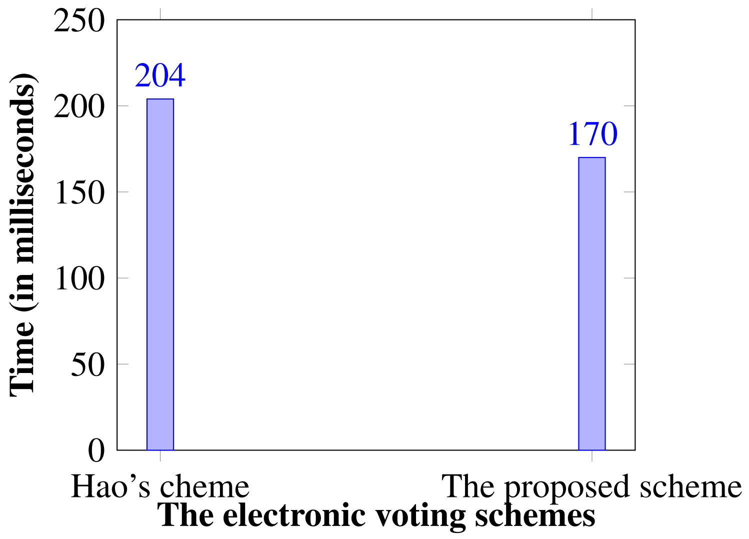


Figure 3.2: The total running time of each voter comparisons between the new solution and Hao’s scheme.

Next, it can be seen in Figure 3.3, the total running time of the voting server in both solutions is nearly linearly related to the number of voters n. Especially, the total running time of the voting server in the new solution is much lower than that in Hao’s scheme. For instance, in the case the number of voters is up to 10000, it only takes the voting server of the new solution 17.07 minutes while that time in Hao’s scheme is 28.46 minutes. This is understandable, because the server in the new solution only computes two share public points for all voters, while the server in Hao’s scheme must compute each public point for each specific voter.

Thus, it can be stated that the proposed electronic voting scheme is efficient and practical.

3.2. An efficient and practical solution for privacy-preserving Naive Bayes classification in the horizontal data setting

This section presents an efficient and practical solution for privacy-preserving Naive Bayes classification in the horizontally distributed data setting. This proposal

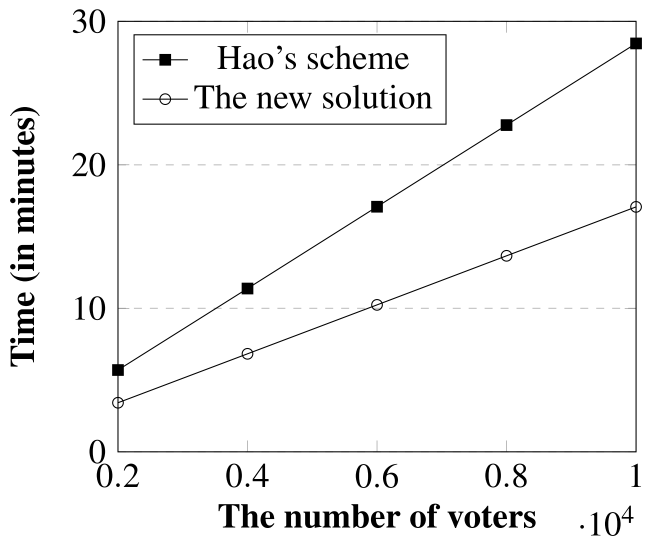


Figure 3.3: The voting server's total running time comparisons between the new solution and Hao's scheme.

of the thesis has related to **Publications** 4 **and** 5.

3.2.1. Introduction

Recently, the advancement of web applications or information systems has created a large amount of data. Based on such data, machine learning models have been constructed to solve complex problems in various fields. Nevertheless, in many cases, processes of machine learning can violate private or sensitive information into data, such as users' political opinions, or patients' diseases, or customers' income. Hence, privacy preservation issues for machine learning techniques have attracted much attention from the researchers [60, 91–96]. The goal of privacy-preserving machine learning (PPML) methods is to build machine learning models while still securely protecting private and sensitive information existing in data.

In essence, a PPML solution should have the following important properties:

- **Accuracy**: a PPML technique should output the same model with the traditional one when built from the same dataset of input vectors.
- **Privacy**: no one knows each holder's data, but himself/herself.
- **Efficiency**: a PPML solution should be efficient enough to be applied in prac-

tice.

Up to now, PPML solutions have been investigated for both centralized and distributed (i.e. horizontal, vertical, and arbitrary) data models. Such solutions are often based on the cryptography, randomization, or hybrid approaches.

- *Randomization approach*: such PPML techniques often hide private and sensitive information by adding noise (e.g., differently privacy technique [97]) to the original data or transforming the original data into the random one. Hence, such PPML solutions can be employed in both data models, and they are quite efficient. Nevertheless, they have a trade-off between privacy and accuracy properties [33, 34, 47, 98].
- *Cryptography-based approach*: PPML solutions based on cryptography techniques often employ SMC protocols based on homomorphic cryptosystems, for example, ElGamal encryption [27], Paillier cryptosystem [28], and Gentry's scheme [99]. As a result, such PPML solutions can securely protect each data holder's privacy without the loss of accuracy. Nevertheless, their performance is often poor.
- *Hybrid approach*: PPML solutions based on both randomization and cryptography techniques have been created to balance the properties (i.e. accuracy, privacy, and efficiency).

According the security aspect, a PPML solution can follow either the semi-honest model or the malicious one [1]. It is assumed in the semi-honest model that the parties comply the computational rules, but there are some corrupted parties who want to know the other participants' private data. In contrast, a participant can do arbitrary behaviors in the malicious model. As a result, design of PPML solutions secure in this model is much more expensive and complex than the semi-honest one. Hence, PPML solutions for the common semi-honest model have been investigated much more than the malicious one.

This study focuses on privacy preservation problem for private and sensitive data used to build the Naive Bayes classifier which is a powerful machine learning technique for predicting unlabeled samples. This section investigates privacy-preserving Naive Bayes classification (`PPNBC`) solutions in the semi-honest model that are used for the horizontal data setting. It can be seen that this scenario is popular in practice. For an instance, a retailer B wants to suggest new services/products based on shopping bills collected from its customers, while they disclose nothing about their data. Another case that a sociologist researches social media problems, in which several online survey questions are provided to some people. However, they do not want to directly reveal their viewpoints because of privacy concerns.

Up to now, a lot of `PPNBC` solutions for the horizontal setting have been proposed, such as [23, 33, 66, 100, 101]. Such solutions are based on the cryptography-based or hybrid approaches. The solutions using cryptography techniques [33, 102, 103] can preserve honest data holders' privacy and ensure classification models' accuracy. Nevertheless, the performance of these solutions is quite poor, because their communication and computational costs are expensive. In contrast, the solutions following the hybrid approach [23,37,104] need to have a trade-off between privacy and accuracy. Thus, it is necessary to create new efficient `PPNBC` solutions having a high level of security, as well as ensuring the accuracy property.

In this section, the thesis chooses the cryptography-based approach. Nevertheless, unlike the existing solutions that often sequentially compute each frequency value used for constructing the `PPNBC` classifier, the new proposal is based on the secure multi-sum computation protocol (presented in Section 2.3.3) that has the capability to obtain all necessary frequency values only in one round of computation. Consequently, the proposed `PPNBC` classifier for the horizontal setting has the following advantages:

- The proposed solution has the capability to securely protect each data holder's privacy, as well as guaranteeing classification models's accuracy. Additionally, any trusted party is not assumed in this solution.
- The proposed solution uses a practical model including data holders and a unique miner who wishes to construct Naive Bayes classifiers by collaborat-

ing with data providers. Furthermore, the proposed solution requires each tuple of data providers to establish no communication channel. Such advantages make the proposed `PPNBC` solution suitable for the distributed data setting.

- The proposed solution's each data provider only executes one round of communication and computation without the key-agreement stage. Hence, it is appropriate to web applications in which each data provider only needs to submit his/her messages once.
- The proposed solution's performance is quite high, because each data provider only spends a small amount of communication and computational cost performing his/her tasks. To prove the proposed solution's applicability and efficiency, the new `PPNBC` solution is deployed for spam short text-messages detection problem using the real data set at Kaggle[1] under privacy constraints.

3.2.2. Related work

Until now, a large number of `PPNBC` solutions have been propounded to be widely applied in a variety of fields, such as malware detection system [108], medical data analytics [103,106,107], or recommendation system [105]. This section provides a review of typical `PPNBC` solutions closely related to the thesis's study.

It is widely known that the first `PPNBC` solution was propounded in the work [100] (another version found in [109]). The authors in [100,109] used the simple secure sum protocol [6,44] to build Naive Bayes classifiers without clearly seeing the parties' private data. Unfortunately, the simple secure sum protocol is weak, so the Naive Bayes classification method [100] cannot securely protect the participants' privacy.

Based on the ElGamal encryption [27], Yang et al. [33] described a privacy-preserving Naive Bayes classification for the fully distributed data setting. The solution of Yang et al. [33] can securely protect the participants' privacy. Neverthe-

[1]`https://www.kaggle.com/uciml/sms-spam-collection-dataset`

less, this solution requires the participants to perform a privacy-preserving frequency computation protocol multiple times. Hence, the PPNBC solution [33] has poor performance.

Yi and Zhang proposed a privacy-preserving solution [47] for Naive Bayes classification technique on distributed dataset using the two semi-trusted mixers model [110], in which each data holder tries to encrypt multiple private values into a unique ciphertext of Paillier encryption [28] and then sends these ciphertexts to the mixers. This setting helps the solution [47] to obtain high performance, but the model using the non-colluding mixers in [47] violates data holders' privacy. Furthermore, the semi-trusted model of Yi and Zhang is different from the one in this study.

Huai et al. [104] propounded a PPNBC solution based on performing a privacy-preserving aggregation protocol [48] on noise-added data multiple times. Moreover, this classification method uses a trusted third party to manage the necessary secret parameters. As a result, the solution [104] has a trade-off between privacy and accuracy properties, and its cost is also pricey.

Based on Gentry's scheme [99], Li et al. [111] built privacy-preserving outsourced Naive Bayes classification for cloud computing model to obtain the following objectives: secure data outsourcing, secure classification model, and secure prediction result. In the training phase of [111], the evaluator S needs to employ an optimized approximation. Consequently, classification models' accuracy cannot be guaranteed in this proposal. Additionally, because Gentry's encryption [99] has extremely high computational complexity [112,113], the cost for deploying the solution [111] is quite expensive.

P. Li et al. [37] and T. Li et al. [23] have introduced two PPNBC solutions based on public-key infrastructures (i.e. Paillier encryption and DD-PKE, respectively) and differential privacy methods to privately protect the providers' data. Hence, these solutions need to have a trade-off between the providers' privacy and classification models' accuracy. Additionally, these solutions require the providers to use high computational cost for executing the necessary cryptography tasks.

Especially, a series of privacy-preserving Naive Bayes classification methods [23, 37, 47, 101, 102, 114] assumed the existing of several non-colluding participants. Unfortunately, the problem of collusion always exists in any computational model (even ideal model) [1,3]. As a result, these solutions [23,37,47,101,102,114] have low level of security.

Skarkala et al. [66] proposed a privacy-preserving Naive Bayesian classification technique based on a multi-candidate election scheme for the vertically and horizontally partitioned database models. By employing Paillier cryptosystem [28], the holders are protected against popular attacks. Furthermore, to improve the security, the parties in [66] are authenticated by a certification authority. Nevertheless, it can be seen in the third phase of Skarkala et al.'s solution that each data holder is required to share the private frequencies of his/her dataset for the miner.

In summary, the existing `PPNBC` solutions have many disadvantages, such as poor performance, low privacy level, or a trade-off between privacy and accuracy properties. This thesis proposes an efficient privacy-preservation solution for Naive Bayes classification problem in the horizontal data model that can ensure the classification model's accuracy and securely protect data providers' privacy.

3.2.3. Preliminaries

3.2.3.1. Naive Bayes classification

To decide that which of the class labels $\{\tau_1, \tau_2, ..., \tau_k\}$ is the predicted label of a new instance $X = (x_1, x_2, ..., x_m)$ (denoted as τ_y), a Naive Bayes classifier is constructed as follows:

$$\begin{aligned} y = & \operatorname*{argmax}_{j=1,\dots,k} \log\{p[j] \cdot \prod_{i=1}^{m} p[i,j]\} \\ = & \operatorname*{argmax}_{j=1,\dots,k} \{\log p[j] + \sum_{i=1}^{m} \log p[i,j]\} \end{aligned} \tag{3.2.2}$$

in which:

- The occurring probability of the class τ_j is $p[j]$.

- The conditional probability of the attribute value x_i given the class τ_j is $p[i,j]$.

Particularly, let $n, n[j], n[i,j]$ be the numbers of samples, samples that their class label is τ_j, and samples that their i^{th} attribute is x_i and their class label is τ_j, respectively, then the following equations are formulated:

$$p[j] = \frac{n[j]}{n} \tag{3.2.3}$$

$$p[i,j] = \frac{n[i,j]}{n[j]} \tag{3.2.4}$$

$$y = \operatorname*{argmax}_{j=1,...,k}\{\log\prod_{i=1}^{m} n[i,j] - \log(n[j]^{m-1}) - \log n\} \tag{3.2.5}$$

Thus, the miner builds Naive Bayes classifier using Equation 3.2.5 needs to obtain the frequency values $n[j], n[i,j]$ in which $i \in \{1,...,m\}$, $j \in \{1,...,k\}$. In the case of the horizontal partition scenario within privacy constraints, such values are often computed by performing privacy-preserving frequency or secure sum computation protocols multiple times, where each data holder uses the statistic values extracted from his/her dataset that are considered as his/her private inputs.

3.2.3.2. Problem of privacy-preserving Naive Bayes classification

The computational model (as illustrated in Figure 3.4) includes:

- *Input*: np datasets of labeled records $\{IV_1,...,IV_{np}\}$, in which each IV_i is privately kept by one data owner U_i ($i \in \{1,...,np\}$).

- *Output*: a Naive Bayes classifier that is constructed from the union dataset $\bigcup_{i=1}^{np} IV_i$.

Hence, the output of this problem is called the privacy-preserving Naive Bayes classifier, and the term "`privacy`" here is to ensure each data provider privately keeps his/her dataset without sharing anyone. It can be seen that this computational model is practical.

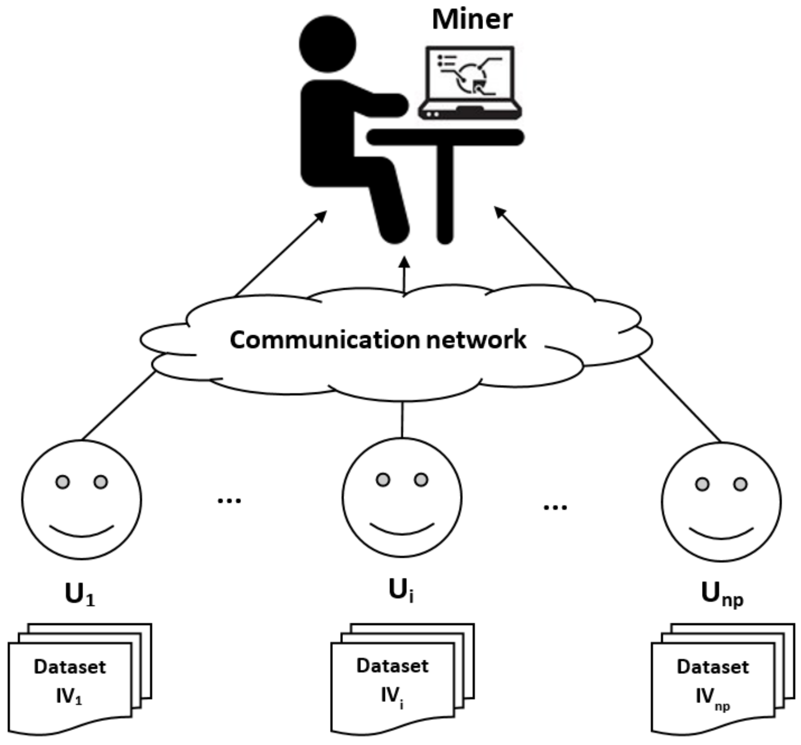


Figure 3.4: The horizontally distributed computing model.

In essence, the miner desires to obtain the set OUT of frequency values summed up from the union dataset $\bigcup_{i=1}^{np} IV_i$. More particularly, for the horizontal partition model, the main challenge of the privacy-preserving Naive Bayes classification problem is to securely compute the set OUT's frequency values ($|OUT|$ denoted as ns) from np sets of ns local frequency values (denoted as $V_{i_{i=1,\dots,np}}$) privately kept by np data owners $\{U_1, \dots, U_{np}\}$. Hence, by employing the efficient and secure multi-sum computation protocol described in Chapter 2, this study obtains a new PPNBC solution that can securely protect the data owners' privacy and ensure the classification model's accuracy. Furthermore, this design also enhances the performance of the proposed solution.

3.2.3.3. Definition of security

Because the proposed PPNBC solution follows the semi-honest model [3], this section states the following definition of security.

Definition 3.2.1. *Assume that each data owner U_i holds a set of private keys Prv_i*

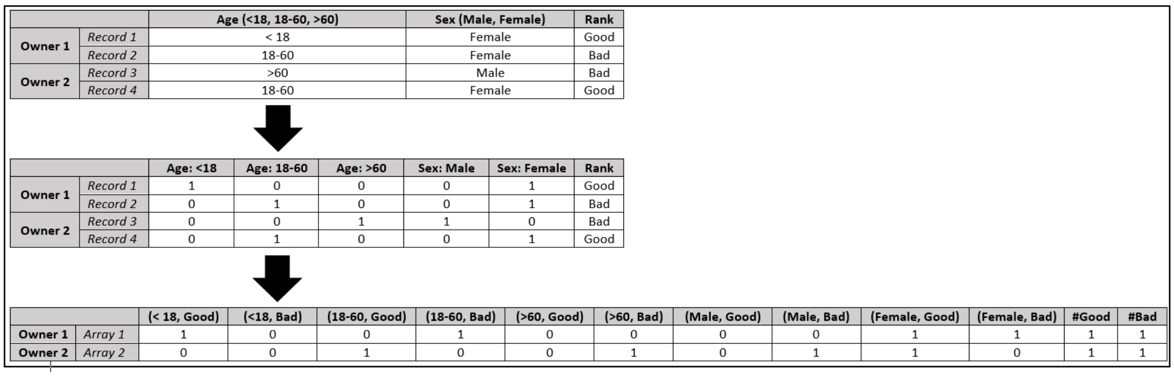
and the set of corresponding public keys Pub_i. A `PPNBC` *protocol protects each data holder's privacy against t corrupted parties and the miner in the common semi-honest model if, for $\forall I \subseteq \{1,2,...,np\}$ ($\|I\| = t$), there exists a probabilistic polynomial-time algorithm $\mathbb{M}$ such that*

$$\{\mathbb{M}(OUT, [V_i, Prv_i]_{i\in I}, [Pub_j]_{j\notin I})\} \overset{c}{\equiv} \{view_{miner,\{U_i\}_{i\in I}}([V_i, Prv_i]_{i=1}^{n})\}$$

where $\overset{c}{\equiv}$ is computational indistinguishability.

3.2.4. New privacy-preserving Naive Bayes classifier for the horizontal partition data setting

In the problem mentioned in Section 3.2.3, suppose that n is total number of records of the datasets $\{IV_1, ..., IV_{np}\}$ and each IV_i ($i \in \{1,...,np\}$) has the structure consisting of na independent attributes $\{A_1, ..., A_{na}\}$ with one class label attribute where A_j has a determined set of k_j values $\{a_j^1, ..., a_j^{k_j}\}$, and each label belongs to $\{L_1, ..., L_{nc}\}$. Hence, each data record is then transformed into a $(nd+1)$ dimensions-vector $(nd = \sum_{j=1}^{na} k_j)$. By aggregating such vectors, each data provider U_i achieves an array of ns private values $\{v_i^1, ..., v_i^{ns}\}$ with $ns = nc + \sum_{j=1}^{na} nc * k_j$. Figure 3.5 illustrates the above transformation.



		Age (<18, 18-60, >60)	Sex (Male, Female)	Rank
Owner 1	*Record 1*	< 18	Female	Good
	Record 2	18-60	Female	Bad
Owner 2	*Record 3*	>60	Male	Bad
	Record 4	18-60	Female	Good

		Age: <18	Age: 18-60	Age: >60	Sex: Male	Sex: Female	Rank
Owner 1	*Record 1*	1	0	0	0	1	Good
	Record 2	0	1	0	0	1	Bad
Owner 2	*Record 3*	0	0	1	1	0	Bad
	Record 4	0	1	0	0	1	Good

		(< 18, Good)	(<18, Bad)	(18-60, Good)	(18-60, Bad)	(>60, Good)	(>60, Bad)	(Male, Good)	(Male, Bad)	(Female, Good)	(Female, Bad)	#Good	#Bad
Owner 1	*Array 1*	1	0	0	1	0	0	0	0	1	1	1	1
Owner 2	*Array 2*	0	0	1	0	0	1	0	1	1	0	1	1

Figure 3.5: An example of data transformation.

In addition, each data owner also computes the number nk of necessary keys following the equation 2.2.14.

In brief, to train Naive Bayes classification models, the miner needs to obtain the sum values $\{\sum_{i=1}^{np} v_i^1, ..., \sum_{i=1}^{np} v_i^{ns}\}$ of the set OUT in a secure way, where $\sum_{i=1}^{np} v_i^j$ ($j \in \{1, ..., ns\}$) is viewed as a frequency value extracted from the union dataset $\bigcup_{i=1}^{np} IV_i$. Hence, by employing the secure multi-sum computation algorithm propounded in Chapter 2, this study proposes a new privacy-preserving Naive Bayes classifier as illustrated in Protocol 3.2.

Before starting the new solution, the miner and the participants also agree to use the cryptographic parameters (p, q, g) as shown in Section 2.3.3.2. The proposed `PPNBC` solution built from the secure multi-party computation protocol is presented as follows.

Protocol 3.2: An efficient solution based on secure multi-party computation protocol for privacy-preserving Naive Bayes classification

```
1. Keys preparation: The data owners $U_i$ do
forall $i$ where $1 \le i \le np$ do
    forall $j$ where $1 \le j \le nk$ do
        $Prv_i^j = Random(1, q-1)$
        $Pub_i^j = g^{Prv_i^j}$
        Sends to Miner: $Pub_i^j$
    end
end
```

```
2. Shared public keys computation: Miner does
forall $j$ where $1 \le j \le nk$ do
    $Pub^j = 1$
    forall $i$ where $1 \le i \le np$ do
        $Pub^j = Pub^j * Pub_i^j$
    end
    Sends to all data owners: $Pub^j$
end
```

```
3. Data submission: The data owners $U_i$ do
forall $i$ where $1 \le i \le np$ do
    $j = 1$
    forall $t$ where $1 \le t \le nk-1$ do
        forall $k$ where $t+1 \le k \le nk$ do
            $p_i^j = g^{v_i^j}(Pub^t)^{Prv_i^k}(Pub^k)^{q-Prv_i^t}$
            if $j == ns$ then
                break
            else
                $j{+}{+}$
            end
        end
    end
    Sends to Miner: $p_i^j$
end
```

```
4. Results Extraction: Miner does
forall $j$ where $1 \le j \le ns$ do
    $K^j = 1$
    forall $i$ where $1 \le i \le np$ do
        $K^j = K^j * p_i^j$
    end
end
Solves the problems $g^{Sum_j} = K^j$
($Sum_j \in \{0, 1, ..., np\}$, $j \in \{1, ..., ns\}$) by
performing the brute-force algorithm once
```

3.2.5. Privacy analysis

To show that each data holder's privacy is protected in the new solution presented in Protocol 3.2, the following theorem is stated and proven.

Theorem 3.2.1. *Protocol 3.2 for Naive Bayes learning securely protects each data holder's privacy against corrupted participants colluding with the miner.*

Proof. As discussed above, the new `PPNBC` solution shown in Protocol 3.2 is created by ns secure multi-sum protocols described in Chapter 2. Furthermore, the thesis proved in Section 2.3 that the new multi-sum computation protocol is secure in the common semi-honest model. Hence, according to Theorem 2.1, the new `PPNBC` solution can securely guarantee each data provider's privacy, even if the miner controls up to $(np-2)$ participants. □

3.2.6. Accuracy analysis

This section proves that the classifier created by the new proposal is equal to that built by the traditional Naive Bayes classification technique when constructed from the same dataset of input vectors.

Theorem 3.2.2. *Protocol 3.2 for Naive Bayes learning ensures the classification model's accuracy.*

Proof. Indeed, as discussed above, using the secure multi-sum protocol proposed in Chapter 2 to obtain the frequency values in the horizontal partition scenario is equivalent to calculating these frequencies in the centralized data model. Additionally, the correctness of the proposed secure multi-sum computation protocol is proved in Section 2.3.3.4. Thus, the new `PPNBC` solution guarantees the classification model's accuracy. □

3.2.7. Experimental evaluation

To show the proposed `PPNBC` solution's advantages, this section compares the experimental results of this proposal with the ones of three typical privacy-preserving

Naive Bayes classification solutions: the solution of Yang et al. in [33], the solution using Hao et al.'s protocol [13] to compute frequency values, and the private Naive Bayes classifier of Hien et al. in the fifth publication (named as Yang's solution, Hao's-based solution, and Hien's solution, respectively).

The thesis chooses the above solutions for the comparisons because they also obtain a high level of privacy and can ensure the classification model's accuracy. Considering each compared solution, the running time of all four phases are reviewed. Recall that np, ns are the numbers of data providers and sum values in the compared solutions, and the number nk of tuples of private & public keys in each solution is calculated from ns.

3.2.7.1. Experimental setting

This section studies a spam messages detection problem using Naive Bayes classification method within privacy constraints as follows:

- *Input*: a set of short ham/spam messages, in which each subset is privately owned by a unique user.
- *Output*: spam messages detection tool based on the Naive Bayes classifier that has built from the above dataset.

To solve this, the experiments are implemented on the dataset of $5,000$ samples randomly chosen from the spam short text-messages collection dataset, publicly available at Kaggle as mentioned above. It is supposed there are 250 users involving in the experiments where each data provider possesses 20 messages (i.e. 14 and 6 ones for training and testing phases, respectively). Each data provider is also shared the dictionary of tokens D (see in Section 3.2.7.2).

Next, the average running time of the phases in each solution is measuring by the available library of programming language. Additionally, the accuracy of the output classification model is also discussed in detail.

For the parameters, the prime numbers p, q are $2048, 256$ bits length, respectively. Moreover, all experiments are implemented in the Python language of Anaconda environment on the Lenovo Thinkpad X280 laptop.

3.2.7.2. Data preparation

i. Data pre-processing

The dataset used in this work is a collection of mobile English text messages, labeled as ham (legitimate) or spam. This dataset has been collected from several free-for-research sources:

- Grumbletext Web site: a public UK forum about spam short text-messages (425 spam messages). [2]
- A subset containing $3,375$ short text-messages randomly chosen from the NUS short-messages corpus (NSC) of the National University of Singapore [3].
- A list of 450 ham messages collected from the University of Birmingham[4].
- Spam Corpus v.0.1 Big containing $1,002$ ham messages and 322 spam messages. [5]

Each data sample includes two elements: the content of the message and its label (ham or spam). Each message's content is preprocessed by removing rare words, common words, or punctuation, and lowercasing all of the tokens. For convenience, the thesis only picks $5,000$ samples from the above dataset. Some statistics information of the pre-processed dataset is presented in Table 3.1.

ii. Features encoding and model setup

Basically, when building text classification model based on the Naive Bayes algorithm, each message's content is often transformed into a vector of features using TF-IDF vectorization Bag-of-Words (BOW) techniques. Nevertheless, with the assumption of existing privacy constraints where whole corpus is not available because

[2]`http://www.grumbletext.co.uk/`

[3]`https://scholarbank.nus.edu.sg/handle/10635/137343`

[4]`https://etheses.bham.ac.uk/id/eprint/253/1/Tagg09PhD.pdf`

[5]`http://www.esp.uem.es/jmgomez/smsspamcorpus/`

Table 3.1: Spam short-messages dataset information

Number of samples	$5,000$
Ham/spam distribution	$4,311/689$
Short-messages max/average length	$211/13.74$

each data provider keeps his/her private short messages. Thus, an external corpus constructed from the public Wikipedia source is used. After that, a BOW model is created that has the k-tokens dictionary D. Here, the thesis sets $k = 1500$.

3.2.7.3. Experimental results

i. Running time

The experimental results are presented in Table 3.2. It can be seen in the column 2 of Table 3.2 that Hien's solution and the new proposal require each data provider about 0.429 seconds for preparing the necessary keys in the phase 1, while those times in Yang's and Hao's-based solution are up to 46.196 and 23.036 seconds, respectively. Continuously, the 3^{rd} column of Table 3.2 illustrates that the miner of Hien's and the proposed solutions uses almost 0.382 seconds for calculating the shared public keys in the phase 2, while those times in Yang's and Hao's-based solutions are 40.506 and 76.287 seconds, respectively. It can be understandable because the numbers of tuples of keys used by each provider in Yang's and Hao's-based solutions are up to 12004 and 6002, respectively, while that number of Hien's solution and the new proposal is only 111.

The column 4 of Table 3.2 shows that the running time for each provider encrypting private values in the phase 3 of Hao's-based solution is smaller than that of others. There is this difference since each data provider in Hao's-based solution only uses one tuple of keys for encrypting one private value. Nevertheless, the total running time for each data provider of the new proposal and Hien's solution (i.e. around 47.163 seconds) is slight more than that in Hao's-based solution (i.e. almost 46.299 seconds), and each data provider's total running time in these three solutions is much

Table 3.2: The running time comparisons among the new proposal and the typical PPNBC solutions on the real dataset (in seconds).

Solutions	**The time**$^{(1)}$	**The time**$^{(2)}$	**The time**$^{(3)}$	**The time**$^{(4)}$
Yang's solution	46.196	40.506	46.602	2808.035
Hao's-based solution	23.036	76.287	**23.263**	81.390
Hien's solution	**0.429**	**0.382**	46.734	81.390
The new solution	**0.429**	**0.382**	46.734	**20.394**

Note: The time$^{(1)}$ for the phase 1, The time$^{(2)}$ for the phase 2.
The time$^{(3)}$ for the phase 3, The time$^{(4)}$ for the phase 4.

smaller than that of Yang's solution (i.e. almost 92.798 seconds).

Lastly, considering the time for the miner extracting the sum values in the last phase, it takes the proposed solution 20.394 seconds, and this time is much smaller than the one in both Hao's-based and Hien's solutions (i.e. about 81.390 seconds). The miner of Yang's solution uses up to 2808 seconds for extracting the sum values.

Especially, regarding all types of execution time in the compared PPNBC solutions, it is easy to know that the new proposal and Hien's solution consume the same amount in the first three phases. This is understandable, because the proposed PPNBC solution's building block is the new secure multi-sum computation protocol. The largest difference between Hien's solution and the new proposal is in the last phase. Particularly, as mentioned before, the miner of the new proposal solves ns discrete logarithm problems $g^{Sum_j} = K^j$ ($Sum_j \in \{0, 1, ..., np\}$, $j \in \{1, ..., ns\}$) by running the brute-force algorithm one time, because they have the same small space $\{0, 1, ..., n\}$ of solutions.

As a result, the above results prove that the proposed PPNBC solution has more advantages than the others. Thus, among the typical PPNBC solutions, the new one is the most suitable solution for practical applications.

ii. Classification accuracy discussion

It is recalled that the aim of the proposed solution is to not only protect each data provider's privacy but also guarantee the accuracy property when compared with the traditional method implementing on a dataset. These requirements has been proven in the above sections.

In the experiments, the model outputted from the proposed solution obtains 0.97 in accuracy, 0.92 in balanced_accuracy, and 0.93 in F1-score. These ratings are equal to the ones of the traditional Naive Bayes classifier implemented on the same dataset.

When compared with the normal Naive Bayes classifier (which achieves $0.99, 0.96, 0.97$ in accuracy, balanced_accuracy, and F1-score, respectively), it is clear that the data transformation process has slightly impacted on the classification model's accuracy. This is understandable, because each record of the dataset has not been transformed into an input vector yet. Hence, privacy preservation problems working on such datasets (e.g. spam short-messages detection) must face the above issue. Nevertheless, the new proposal's accuracy is still high.

iii. Deployment issues of the proposed `PPNBC` *solution*

In practice, the deployment of the proposed `PPNBC` solution is quite simple. Firstly, this solution (also including the public dictionary D) needs to be packaged into a mobile application (e.g. iOS or Android app) to be installed on smartphones. When receiving a new `SMS`, the application uses the dictionary D to locally transform it into a private binary vector. Next, the application predicts the label (ham or spam?) of this vector based on the parameters of Naive Bayes classification model.

3.3. Conclusion

In this chapter, based on the new protocols presented in Chapter 2, the thesis has developed new solutions for two practical problems. In the first one, the thesis constructed the secure voting scheme in the end-to-end decentralized environment.

For the second problem, a privacy-preserving Naive Bayes classifier is built for the horizontal partition model. The necessary experiments have been run, and the experimental results proved that the new proposals not only satisfy the requirements of practical problems, but also outperform the previous solutions. It is also additionally analyzed that the proposed protocols in this thesis assume that all participants do not abort the protocols prematurely that cannot be avoided even in the malicious model. Nevertheless, the proposed protocols and solutions have the capability to prevent up to $(n-2)$ colluding participants, in which n is the total number of parties.

CONCLUSION AND FUTURE WORK

The thesis has proposed three new secure multi-party sum computation protocols and developed new solutions based on these protocols for two practical applications. For each proposal, the thesis analyzed the security aspect as well as evaluated the performance. The results of the thesis can be summarized as follows:

- In the first work, the thesis has propounded three novel secure multi-party sum computation protocols that are the elliptic curve analog of the ElGamal cryptosystem-based protocol for privacy-preserving frequency computation in fully distributed setting, the secure multi-party sum computation protocol without pre-establishing any authenticated/secure channel, and the secure multi-sum computation protocol in one round of computation. The proposed protocols are secure in the semi-honest model. Furthermore, these new proposals are efficient enough to be implemented in real-life applications.
- In the second work, the thesis has developed new solutions based on the proposed protocols for two practical applications that are the secure end-to-end e-voting scheme without pre-establishing secure/authenticated channels, and an efficient and practical method for privacy-preserving Naive Bayes classification in the horizontally distributed data setting. The security analysis results have proved that the new solutions satisfy the requirements of applications. Besides, the experimental evaluations has showed the new proposals' efficacy, so they can be implemented in practice.

Next, the thesis discusses some potential issues of the general SMC field in the future.

- Firstly, as mentioned in Chapter 1, the birth of SMC field was initiated from distributed computing scenarios and practical problems. As a result, based on new distributed computing scenarios or the requirements of practical problems, new secure multi-party computation protocols are necessary to be investigated by research community.

- Secondly, secure multi-party computation protocols should be based on modern cryptography (e.g. post-quantum cryptographic techniques) with the aim of preventing potential attacks on the next generation of computing that is coming closer to us.
- Thirdly, because of SMC protocols' applicability in various domains, such protocols should be considered for the implementation in real-life applications. This helps not only distributed computing tasks to be performed more easily, but also the participants' safety to be protected.

BIBLIOGRAPHY

[1] Yehuda Lindell and Benny Pinkas. Secure Multiparty Computation for Privacy-Preserving Data Mining. *Journal of Privacy and Confidentiality*, 1:59–98, 2009.

[2] Chuan Zhao, Shengnan Zhao, Minghao Zhao, Zhenxiang Chen, Chong-Zhi Gao, Hongwei Li, and Yu-an Tan. Secure Multi-Party Computation: Theory, practice and applications. *Information Sciences*, 476:357–372, 2019.

[3] Oded Goldreich. Basic Applications. In *Foundations of Cryptography*, volume II. Cambridge University Press, 2004.

[4] Andrew C Yao. Protocols for Secure Computations. In *Proceedings of 23rd Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE, 1982.

[5] Chuan Zhao, Shengnan Hao, Bo Zhang, Zhongtian Jia, Zhenxiang Chen, and Mauro Conti. Secure comparison under ideal/real simulation paradigm. *IEEE Access*, 6:31236–31248, 2018.

[6] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y. Zhu. Tools for Privacy Preserving Distributed Data Mining. *ACM SIGKDD Explorations Newsletter*, 4:28–34, 2002.

[7] Shintaro Urabe, Jiahong Wang, Eiichiro Kodama, and Toyoo Takata. A High Collusion-Resistant Approach to Distributed Privacy-preserving Data Mining. *Information and Media Technologies*, 48:104–117, 2007.

[8] Zhu Youwen, Huang Liusheng, Yang Wei, and Yuan Xing. Efficient Collusion-Resisting Secure Sum Protocol. *Chinese Journal of Electronics*, 20:407–413, 2011.

[9] Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikainen. On Pri-

vate Scalar Product Computation for Privacy-Preserving Data Mining. In *Proceedings of International Conference on Information Security and Cryptology*, pages 104–120. Springer, 2004.

[10] Changyu Dong and Liqun Chen. A Fast Secure Dot Product Protocol with Application to Privacy Preserving Association Rule Mining. In *Proceedings of The 18th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2014.

[11] H. Tran Duc, Wee Keong Ng, Hoon Wei Lim, and Hai-Long Nguyen. An Efficient Cacheable Secure Scalar Product Protocol for Privacy-Preserving Data Mining. In *Proceedings of the 13th International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*. Springer-Verlag, 2011.

[12] F Hao, P.Y.A Ryan, and P Zielin´ski. Anonymous voting by two-round public discussion. *IET Information Security*, 4:62 – 67, 2010.

[13] Feng Hao, Matthew N. Kreeger, Brian Randell, Dylan Clarke, Siamak F. Shahandashti, and Peter Hyun-Jeen Lee. Every Vote Counts: Ensuring Integrity in Large-Scale Electronic Voting. *USENIX Journal of Election Technology and Systems*, 2:1–25, 2014.

[14] Peter Bogetoft, Dan Lund Christensen, Ivan Damgard, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Jesper Buus, Jakob Pagter, Michael Schwartzbach, and Tomas Toft. Secure Multiparty Computation Goes Live. In *Proceedings of 2009 Financial Cryptography and Data Security*, pages 325–343. Springer, 2009.

[15] Galois. Jana: Private Data as a Service, January 2020.

[16] Dan Bogdanov, Riivo Talviste, and Jan Willemson. Deploying Secure Multi-Party Computation for Financial Data Analysis. In *Proceedings of 2012 Financial Cryptography and Data Security*. Springer, 2012.

[17] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. On deploying

secure computing: Private intersectionsum-with-cardinality. In *Proceedings of 2020 IEEE European Symposium on Security and Privacy*. IEEE, 2020.

[18] Azer Bestavros, Andrei Lapets, and Mayank Varia. User-centric distributed solutions for privacy-preserving analytics. *Communications of the ACM*, 2:37–39, 2017.

[19] Jian Liu, Mika Juuti, Yao Lu, and N Asokan. Oblivious Neural Network Predictions via MiniONN Transformations. In *Proceedings of 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 619–631. ACM, 2017.

[20] D Malkhi, N Nisan, and B Pinkas. Fairplay – A Secure Two-Party Computation System. In *Proceedings of 13th USENIX Security Symposium*. USENIX Association, 2004.

[21] Dongsheng Li, Chao Chen, Qin Lv, Li Shang, Yingying Zhao, Tun Lu, and Ning Gu. An algorithm for efficient privacy-preserving item-based collaborative filtering. *Future Generation Computer Systems*, 55:311–320, 2016.

[22] Shagufta Mehnaz, Gowtham Bellala, and Elisa Bertino. A secure sum protocol and its application to privacy-preserving multi-party analytics. In *Proceedings of 22nd The ACM Symposium on Access Control Models and Technologies (SACMAT)*. ACM, 2017.

[23] Tong Li, Jin Li, Zheli Liu, Ping Li, and Chunfu Jia. Differentially private Naive Bayes learning over multiple data sources. *Information Sciences*, 444:89–104, 2018.

[24] Daniele Croce, Fabrizio Giuliano, Ilenia Tinnirello, and Laura Giarré. Privacy-preserving overgrid: Secure data collection for the smart grid. *Sensors*, 20, 2020.

[25] M.J. Atallah, H.G. Elmongui, V. Deshpande, and L.B. Schwarz. Secure Supply-Chain Protocols. In *Proceedings of International Conference on E-Commerce*. IEEE, 2003.

[26] Taeho Jung, Junze Han, and Xiang-Yang Li. PDA: Semantically secure time-series data analytics with dynamic subgroups. *IEEE Transactions on Dependable and Secure Computing*, 15:260–274, 2018.

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

[28] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 1999.

[29] Apple. Password monitoring.

[30] Juan Ramón Troncoso-Pastoriza, Stefan Katzenbeisser, and Mehmet Celik. Privacy preserving error resilient dna searching through oblivious automata. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, page 519–528. Association for Computing Machinery, 2007.

[31] Quang Do, Ben Martini, and Choo Kim-Kwang Raymond. The role of the adversary model in applied security research. *Computers & Security*, 81:156–181, 2019.

[32] Yehuda Lindell and Benny Pinkas. Privacy Preserving Data Mining. In *Proceedings of 20th Annual International Cryptology Conference*. Springer, 2000.

[33] Zhiqiang Yang, Sheng Zhong, and Rebecca N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 92–102. SIAM, 2005.

[34] Luong The Dung and Ho Tu Bao. Privacy Preserving Frequency Mining in 2-Part Fully Distributed Setting. *IEICE Transactions on Information and Systems*, 93:2702–2708, 2010.

[35] Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, and Lihua Wang. Input and

Output Privacy-Preserving Linear Regression. *IEICE Transcations on Information and Systems*, 100:2339–2347, 2017.

[36] Ping Li, Jin Li, Zhengan Huang, Chong-Zhi Gao, Wen-Bin Chen, and Kai Chen. Privacy-preserving outsourced classification in cloud computing. *Cluster Computing*, 21:277–286, 2018.

[37] Ping Li, Tong Li, Heng Ye, Jin Li, Xiaofeng Chen, and Yang Xiang. Privacy-preserving machine learning with multiple data providers. *Future Generation Computer Systems*, 87:341–350, 2018.

[38] Tran Huy Duc. *Speeding up privacy preserving data mining techniques*. Doctor of Philosophy, Nanyang Technological University, 2016.

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

[40] Yehuda Lindell. *How to Simulate It – A Tutorial on the Simulation Proof Technique*. Springer, 2017.

[41] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. CRC PRESS, 2007.

[42] C.P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.

[43] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computatio*, 48:203–209, 1987.

[44] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, 2 edition, 1996.

[45] En Zhang and Yongquan Cai. Collusion–free rational secure sum protocol. *Chinese Journal of Electronics*, 22:563–566, 2013.

[46] Yongkang Luo, Wenjian Luo, Ruizhuo Zhang, Hongwei Zhang, and Yuhui Shi. Robust peer-to-peer learning via secure multi-party computation. *Journal of Information and Intelligence*, 1:341–351, 2023.

[47] Xun Yi and Yanchun Zhang. Privacy-preserving Naive Bayes classification

on distributed data via semi-trusted mixers. *Information Systems*, 34:371–380, 2009.

[48] Elaine Shi, T-H. Hubert Chan, Eleanor Rieffel, Richard Chow, and Dawn Song. Privacy-Preserving Aggregation of Time-Series Data. In *Proceedings of 2011 The Network and Distributed System Security Symposium*. Internet Society, 2011.

[49] Taeho Jung, Xiang-Yang Li, and Meng Wan. Collusion-Tolerable Privacy-Preserving Sum and Product Calculation without Secure Channel. *IEEE Transactions on Dependable and Secure Computing*, 12:45 – 57, 2014.

[50] Amit Datta and Marc Joye. Cryptanalysis of a privacy-preserving aggregation protocol. *IEEE Transactions on Dependable and Secure Computing*, 14:693–694, 2016.

[51] Shahriar Badsha, Xun Yi, and Ibrahim Khalil. A Practical Privacy-Preserving Recommender System. *Data Science and Engineering*, 1:161–177, 2016.

[52] Tran Anh-Tu, Luong The-Dung, Karnjana Jessada, and Van-Nam Huynh. An efficient approach for privacy preserving decentralized deep learning models based on secure multi-party computation. *Neurocomputing*, 422:245–262, 2021.

[53] Yongli Wang, Peichu Hu, and Qiuliang Xu. Quantum secure multi-party summation based on entanglement swapping. *Quantum Information Processing*, 20, 2021.

[54] Kartick Sutradhar and Hari Om. An efficient simulation for quantum secure multiparty computation. *Nature Scientific Reports*, 11, 2021.

[55] Wan-Qing Wu and Ming-Zhe Xie. Quantum secure multi-party summation using single photons. *Entropy*, 25, 2023.

[56] Ning Wang, Xinying Tian, Xiaodong Zhang, and Song Lin. Quantum secure multi-party summation with identity authentication based on commutative encryption. *Photonics*, 10, 2023.

[57] Fan Wu, Jiqiang Liu, and Sheng Zhong. An efficient protocol for private and accurate mining of support counts. *Pattern Recognition Letters*, 30:80–86, 2009.

[58] Feng Hao, Dylan Clarke, Brian Randell, and Siamak F. Shahandashti. Verifiable Classroom Voting in Practice. *IEEE Security & Privacy*, 16:72–81, 2018.

[59] F Hao and P Zieliński. A 2-Round Anonymous Veto Protocol. In *Proceedings of 14th International Workshop on Security Protocols*. Springer, 2006.

[60] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191. ACM, 2017.

[61] J. Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single–server aggregation with (poly)logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2020.

[62] Wei-Ning Chen, Christopher A. Choquette-Choo, Peter Kairouz, and Ananda Theertha Suresh. The fundamental price of secure aggregation in differentially private federated learning. In *Proceedings of the 39th International Conference on Machine Learning*. Curran Associates, Inc., 2022.

[63] J. Henry Bell, Adrià Gascón, Tancrède Lepoint, Baiyu Li, Sarah Meiklejohn, Mariana Raykova, and Cathie Yun. Acorn: Input validation for secure aggregation. In *Proceedings of 32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, 2023.

[64] Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard. Practical Multi-Candidate Election System. In *Proceedings of the 20th annual ACM symposium on Principles of distributed computing*, pages 274–283. ACM, 2001.

[65] Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, and Lihua Wang. Privacy-preserving logistic regression with distributed data sources via homomorphic encryption. *IEICE Transactions on Information and Systems*, 99(8):2079–2089, 2016.

[66] Maria Eleni Skarkala, Manolis Maragoudakis, Stefanos Gritzalis, and Lilian Mitrou. PPDM-TAN: A Privacy-Preserving Multi-Party Classifier. *Computation*, 9:1–25, 2021.

[67] ARTRIM KJAMILJI, ERKAY SAVAS, and ALBERT LEVI. Efficient secure building blocks with application to privacy preserving machine learning algorithms. *IEEE Access*, 9, 2021.

[68] Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *Proceedings of the 38th International Conference on Machine Learning*, pages 5201–5212. Curran Associates, Inc., 2021.

[69] Jianhong Zhang and Chenghe Dong. Privacy-preserving data aggregation scheme against deletion and tampering attacks from aggregators. *Journal of King Saud University – Computer and Information Sciences*, 35:100—-111, 2023.

[70] Xuechao Yang, Xun Yi, Surya Nepal, Andrei Kelarev, and Fengling Han. Blockchain voting: Publicly verifiable online voting protocol without trusted tallying authorities. *Future Generation Computer Systems*, 112:859–874, 2020.

[71] Sangsoo Seol, Hyoseung Kim, and Jong Hwan Park. An efficient open vote network for multiple candidates. *IEEE Access*, 10:124291–124304, 2022.

[72] Nguyen Ngan and Nguyen-An Khuong. A transparent scalable e-voting protocol based on open vote network protocol and zk-starks. In *Proceedings of International Conference on Intelligence of Things*. Springer, Cham, 2023.

[73] Daniel Shanks. Class number, a theory of factorization and genera. In *Proceedings of Symposium of Pure Mathematics*, pages 415–440. AMS, 1971.

[74] Daniel J. Bernstein. Curve25519: New Diffie-Hellman Speed Records. In *Proceedings of the 9th International Conference on Theory and Practice in Public-Key Cryptography*. Springer, 2006.

[75] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC PRESS, 1996.

[76] Jonathan Katz. *Digital Signatures*. Springer, 2010.

[77] David Pointcheval and Jacques Stern. Security Proofs for Signature Schemes. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*. Springer, 1996.

[78] David Pointcheval and Jacques Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of cryptology*, 13, 2000.

[79] Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. RIPEMD-160: A strengthened version of RIPEMD. In *Proceedings of the 3rd International Workshop on Fast Software Encryption*. Springer, 1996.

[80] M. Lepinski and S. Kent. Additional Diffie-Hellman Groups for Use with IETF Standards, 2008.

[81] Aggelos Kiayias and Moti Yung. Self-tallying elections and perfect ballot secrecy. In *Proceedings of International Conference on Practice and Theory of Public-Key Cryptography*. Springer, 2002.

[82] Jens Groth. Efficient maximal privacy in boardroom voting and anonymous broadcast. In *Proceedings of International Conference on Financial Cryptography*, 2004. Springer, 2004.

[83] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proceedings of International Workshop on Security Protocols*, pages 25–35. Springer, 1997.

[84] Markus Jakobsson. Flash mixing. In *Proceedings of the 18th annual ACM symposium on Principles of distributed computing*, pages 83–89. ACM, 1999.

[85] J Paul Gibson, Robert Krimmer, Vanessa Teague, and Julia Pomares. A review of E-voting: the past, present and future. *Annals of Telecommunications*, 71:279–286, 2016.

[86] Ed Gerck, C. Andrew Neff, Ronald L. Rivest, Aviel D. Rubin, and Moti Yung. The business of electronic voting. In *Proceedings of International Conference on Financial Cryptography*. Springer, 2001.

[87] Dimitris A Gritzalis. Principles and requirements for a secure e-voting system. *Computers & Security*, 21:539–556, 2002.

[88] Chin-Ling Chen, Yu-Yi Chen, Jinn-Ke Jan, and Chih-Cheng Chen. A secure anonymous e-voting system based on discrete logarithm problem. *Applied Mathematics & Information Sciences*, 8:2571–2578, 2014.

[89] Syed Taha Ali and Judy Murray. An Overview of End-to-End Verifiable Voting Systems. In *Real-World Electronic Voting*. CRC PRESS, first edition, 2016.

[90] Ben Adida. Helios: Web-based Open-Audit Voting. In *Proceedings of 17th USENIX Security Symposium*, 335-348, 2008. USENIX Association.

[91] Mohammad Al-Rubaie and J. Morris Chang. Privacy-Preserving Machine Learning: Threats and Solutions. *IEEE Security & Privacy*, 17:49–58, 2019.

[92] M.A.P. Chamikara, Bertok P., I. Khalil, D. Liu, and S. Camtepe. Privacy preserving distributed machine learning with federated learning. *Computer Communications*, 171:112–125, 2021.

[93] Georgios A. Kaissis, Marcus R. Makowski, Daniel Rückert, and Rickmer F. Braren. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2:305–311, 2020.

[94] Jun Zhang, ZoeL. Jiang, Ping Li, and Siu Ming Yiu. Privacy-preserving mul-

tikey computing framework for encrypted data in the cloud. *Information Sciences*, 575:217–230, 2021.

[95] Xianfei Zhou, Kai Xu, Naiyu Wang, Jianlin Jiao, Ning Dong, Meng Han, and Hao Xu. A Secure and Privacy-Preserving Machine Learning Model Sharing Scheme for Edge-Enabled IoT. *IEEE Access*, 9:17256 – 17265, 2021.

[96] Ezgi Zorarpacı and Selma Ayşe Özel. Privacy preserving classification over differentially private data. *Wires Data mining and knowledge discovery*, 11:1–20, 2021.

[97] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9:211–407, 2014.

[98] Andy Wang, Chen Wang, Meng Bi, and Jian Xu. A Review of Privacy-Preserving Machine Learning Classification. In *Proceedings of International Conference on Cloud Computing and Security*, pages 671–682. Springer, 2018.

[99] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178. ACM, 2009.

[100] Murat Kantarcıo˘glu, Jaideep Vaidya, and Chris Clifton. Privacy Preserving Naive Bayes Classifier for Horizontally Partitioned Data. In *Proceedings of IEEE ICDM workshop on privacy preserving data mining*. IEEE, 2003.

[101] Zhu Youwen, Xingxin Li, Jian Wang, Yining Liu, and Zhiguo Qu. Practical Secure Naıve Bayesian Classification Over Encrypted Big Data in Cloud. *International Journal of Foundations of Computer Science*, 28:683–703, 2017.

[102] Sangwook Kim, Masahiro Omori, Takuya Hayashi, Toshiaki Omori, Lihua Wang, and Seiichi Ozawa. Privacy-Preserving Naive Bayes Classification Using Fully Homomorphic Encryption. In *Proceedings of the 25th International Conference on Neural Information Processing*, pages 349–358. Springer, 2018.

[103] Alexander Wood, Vladimir Shpilrain, Kayvan Najarian, and Delaram

Kahrobaei. Private naïve Bayes classification of personal biomedical data: Application in cancer data analysis. *Computers in Biology and Medicine*, 105:144–150, 2019.

[104] Mengdi Huai, Liusheng Huang, Wei Yang, Lu Li, and Mingyu Qi. Privacy-preserving Naive Bayes classification. In *Proceedings of International conference on knowledge science, engineering and management*, pages 627–638. Springer, 2015.

[105] Cihan Kaleli and Huseyin Polat. Privacy-preserving Naïve Bayesian classifier-based recommendations on distributed data. *Computational Intelligence*, 31(1):47–68, 2015.

[106] Ximeng Liu, Rongxing Lu, Jianfeng Ma, Le Chen, and Baodong Qin. Privacy-preserving patient-centric clinical decision support system on naïve Bayesian classification. *IEEE Journal of Biomedical and Health Informatics*, 20:655 – 668, 2016.

[107] Xiaoxia Liu, Hui Zhu, Rongxing Lu, and Hui Li. Efficient privacy-preserving online medical primary diagnosis scheme on Naïve Bayesian classification. *Peer-to-Peer Networking and Applications*, 11:334–347, 2018.

[108] Zhaowen Lin, Fei Xiao, Yi Sun, Ma Yan, Cong-Cong Xing, and Jun Huang. A Secure Encryption-Based Malware Detection System. *KSII Transactions on Internet and Information Systems*, 12(4):1799–1818, 2018.

[109] Jaideep Vaidya, Murat Kantarcioglu, and Chris Clifton. Privacy-preserving Naïve Bayes classification. *The VLDB Journal*, 17:879–898, 2008.

[110] Xun Yi and Yanchun Zhang. Privacy-preserving distributed association rule mining via semi-trusted mixer. *Data & Knowledge Engineering*, 63:550–567, 2007.

[111] Ping Li, Jin Li, Zhengan Huang, Chong-Zhi Gao, Wen-Bin Chen, and Kai Chen. Privacy-preserving outsourced classification in cloud computing. *Cluster Computing*, 21:277–286, 2018.

[112] Chong-Zhi Gao, Qiong Cheng, Pei He, Willy Susilo, and Jin Li. Privacy-preserving Naive Bayes classifiers secure against the substitution-then-comparison attack. *Information Sciences*, 444:72–88, 2018.

[113] Chong-Zhi Gao, Jin Li, Shibing Xia, Kim-Kwang Raymond Choo, Wenjing Lou, and Changyu Dong. MAS-Encryption and Its Applications in Privacy-Preserving Classifiers. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–17, 2020.

[114] Xingxin Li, Youwen Zhu, and Jian Wang. Secure Naıve Bayesian Classification over Encrypted Data in Cloud. In *Proceedings of the 10th International Conference on Provable Security*, pages 130–150. Springer, 2016.

[115] Arjen Klaas Lenstra and Hendrik Willem Lenstra Jr. *Algorithms in Number Theory*. Elsevier Science, 1991.

APPENDICES

A. Shanks' baby-step giant-step algorithm

In this section, the thesis present Shanks' algorithm [73,115] that is an efficient method used for solving discrete logarithm problems.

Input: A cyclic group G of order q has a generator g, and an element y

Output: A value x satisfies $g^x = y$

```
$m \leftarrow \lceil \sqrt{q} \rceil$
forall $j$ where $0 \leq j < m$ do
    Computes $g^j$ and store the pair $(j, g^j)$ in a hash table
end
Computes $g^{-m}$
$\beta \leftarrow y$
forall $i$ where $0 \leq i < m$ do
    if $\beta$ is the second component $(g^j)$ of any pair in the hash table then
        Returns $x = i.m + j$
    else
        $\beta \leftarrow \beta.g^{-m}$
    end
end
```

Algorithm 1: Shanks' baby-step giant-step algorithm.

Specifically, if it can be limited the range of value x (i.e., $x \leq n \ll q$), then Shanks' algorithm runs much more efficient.

Simultaneously, there is a variant of this Shanks' algorithm based on the elliptic-curve cryptography described as follows:

Input: An elliptic curve (E) over $(Z)_p$ with a base point G and a point $Y \in E$

Output: A value x satisfies $xG = Y$

```
$m \leftarrow \lceil \sqrt{q} \rceil$
forall $j$ where $0 \leq j < m$ do
    Computes $jG$ and store the pair $(j, jG)$ in a hash table
end
Computes $A = -(mG)$
$B \leftarrow Y$
forall $i$ where $0 \leq i < m$ do
    if $\beta$ is the second component $(jG)$ of any pair in the hash table then
        Returns $x = i.m + j$
    else
        $B \leftarrow B + A$
    end
end
```

Algorithm 2: The variant of Shanks' baby-step giant-step algorithm.

B. Brute-force algorithm

Although the brute-force algorithm is one of the worst methods for solving discrete logarithms, this method is efficient in the case that the solution space of discrete logarithm problems are not too large.

```
Input: A cyclic group $\mathbb{G}$ of order $q$ has a generator $g$, and an element $y \in \mathbb{G}$.
Output: A value $x \in \{0, ..., max\_value\}$ satisfies $g^x = y$.
$K = 1$
forall $i$ where $0 \leq i \leq max\_value$ do
    if $K = y$ then
        output $i$.
    else
        $K = K * g$
    end
end
```

Algorithm 3: Brute-force algorithm for solving discrete logarithms.

PUBLICATION LIST

1. **Duy-Hien Vu**, The-Dung Luong, Tu-Bao Ho, and Chung-Tien Nguyen. Privacy-preserving frequency mining protocol based on elliptic curve ElGamal cryptosystem. *HNUE Journal of Science*, 63:87-96, 2018.

2. **Duy-Hien Vu**, The-Dung Luong, Tu-Bao Ho, and Chung-Tien Nguyen. An Efficient Approach for Electronic Voting Scheme without An Authenticated Channel. In *Proceedings of the 10th International Conference on Knowledge and Systems Engineering*, pages 376-381. IEEE, 2018.

3. **Duy-Hien Vu**, The-Dung Luong, and Tu-Bao Ho. An efficient approach for secure multi-party computation without authenticated channel. *Information Sciences*, 527:356-368, 2020. (SCI/ISI indexed, Scopus Q1)

4. **Duy-Hien Vu**, Trong-Sinh Vu, and The-Dung Luong. An efficient and practical approach for privacy-preserving Naive Bayes classification. *Journal of Information Security and Applications*, 68, 2022. (SCIE/ISI indexed, Scopus Q1)

5. **Vu Duy Hien**, Luong The Dung, and Hoang Duc Tho. An Efficient Solution for Privacy-preserving Naive Bayes Classification in Fully Distributed Data Model. *Journal of Science and Technology on Information Security*, 15:56-62, 2022.