JAIST Repository
https://dspace.jaist.ac.jp/

Title	プライバシーを保存するいくつかの計算問題への効率的なソリューション
Author(s)	桑，応朋
Citation	
Issue Date	2007-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/3571
Rights	
Description	Supervisor:丹　康雄，情報科学研究科，博士

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

Efficient Solutions to Several Privacy Preserving Computation Problems

by

Yingpeng SANG

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Associate Professor Yasuo Tan

School of Information Science
Japan Advanced Institute of Science and Technology

March, 2007

Abstract

In a distributed network, Secure Multiparty Computation (SMC) is always required by the participants who want to compute some functions on their inputs, while ensuring independence of the inputs, correctness of the computation, and that no more information is revealed to a participant other than can be computed from that participant's input and output. In this thesis, we aim at several specific problems in applications of e-bidding, database, and wireless sensor network, all of which belong to the general SMC problem. For these problem, we propose solutions with lower complexities and stronger security than previous results. Specifically, our work is composed of the following four parts:

- We address the problem of Secure Two-party Vector Dominance (STVD) which can be encountered in applications such as multi-commodity private bidding. Alice has a bidding vector $(a_1, ..., a_n)$ and Bob has a price vector $(b_1, ..., b_n)$ for a series of commodities. They would make a deal if there is the dominance relation $\{a_i \geq b_i | i = 1, ..., n\}$, while protecting their privacy on the vectors. Though STVD is a multi-dimensional extension of Yao's millionaire problem which has been extensively researched, it cannot be solved by trivially n rounds of a solution for the millionaire problem, because leaking the ordering of each pair (a_i, b_i) to each other is considered a breach of privacy, when there is no dominance relation. In this thesis we propose an STVD protocol for the semi-honest model, based on an additive homomorphic encryption, and then fix it to be a secure protocol in the malicious model by efficient zero-knowledge proofs. Let K be the length of each element in the vector. By parallel executions in which Alice has K platforms and Bob has one platforms, our protocol for the semi-honest case need less modular exponentiations than a derived solution from [84] and a solution from [4], and our protocol for the malicious case has the same level of computation with the derived solution from [84]. Though our protocols for the two cases need more communication bits, they can be completed by no more than one second in practical applications, and will not decrease the efficiency of our protocols.

- We address the Privacy Preserving Set Intersection (PPSI) problem, in which each party learns no elements other than the intersection of the N private datasets. When datasets are distributed on different sources, finding out their intersection while preserving the privacy of the datasets is a widely required task. We propose an efficient protocol based on a threshold cryptosystem which is additive homomorphic. The protocol is firstly constructed assuming the adversary is semi-honest and controls arbitrary number of parties, then it is extended to resist the malicious behaviors of the adversary. In comparisons with previous results in [36], [62] and [63], our PPSI protocols in the semi-honest and malicious models achieve lower computation and communication costs.

- We address the problem of Privacy Preserving Tuple Matching (PPTM) in a scenario of a horizontally partitioned database among N parties, where each party

holds a share of the database's tuples and all tuples have the same set of attributes. Each party wants to determine whether its tuples match those of other parties on all attributes, under the condition that no party publishes its own tuples for privacy concern. We show that another related problem, Privacy Preserving Threshold Attributes Matching (PPTAM), can also be solved by similar techniques. By experiments in a moderate-scale application, our protocol for PPTM saves about 80% computation time in comparison with previous results, with an increase of bandwidth that can be transferred within a few seconds. Though a solution can be derived from the techniques in [62], we are the first to address the PPTAM problem to our knowledge. Our protocol for PPTAM saves about 71.3% computation time and 75% bandwidth in comparison with the derived solution. Both of our protocols are proved to be secure in the semi-honest model.

- We discuss two conflictive privacy issues in pervasive sensor networks, including the originator privacy and sensing area privacy. More and more sensor networks will be deployed in the place where people are living, studying, and working. These sensor networks will bring us the convenience of accessing information anytime and anywhere, whereas put our voice, motion, or even body temperature under surveillance. Under the circumstances of pervasively deployed sensor networks, people will have a dynamic concern about their privacy. At the same time, sensors will become invisible or should be hidden due to the privacy of themselves. We propose a general scheme for people in the environment of pervasive sensor networks, so that they can be aware of whether they should be alert on their privacy activities. Our scheme employs the STVD protocol we achieve in the first part of work as a building block, and has the characteristics of generality and confidentiality.

In sum, we are the first to propose an efficient STVD protocol in the malicious model, which can be robust against various malicious attacks of powerful adversaries. We propose PPSI protocols in the semi-honest and malicious models which cost less computation time and bandwidth in practical applications than previous results. We propose a PPTM protocol which increase seconds of communication time, but reduce hours of computation time, compared with previous results. We are the first to solve the PPTAM problem and achieve a much more efficient protocol than a solution derived from previous techniques. We are also the first to provide solutions to protect the conflictive privacy issues, originator privacy and sensing area privacy, in pervasive sensor network.

Keywords : privacy preservation, secure computation, homomorphic encryption, zero knowledge proof, distributed database.

Acknowledgments

Firstly I'm deeply grateful to my advisors Professor Hong Shen and Associate Professor Yasuo Tan for their continuous illuminations and invaluable instructions on my work. Professor Hong Shen has provided enough space for me to expand my interest in this work, leaded me to the intrinsic properties of every problem, and enlightened me with systematic methods to do research. Associate Professor Yasuo Tan has given scrupulous care on every difficulty in finalizing the work, and provided precious suggestions in the publications. Without any of them it would probably not be possible to complete this thesis.

My sincere thanks would also be given to Associate Professor Yasushi Inoguchi, the advisor of my sub-theme, for his fruitful guidance in my sub-theme and strict construction on my research manner. I would like to thank Professor Yoichi Shinoda, Associate Professor Atsuko Miyaji, Associate Professor Xavier Defago, and Dr. Akihiro Yamamura for taking time from their busy schedules to serve on the thesis defense committee.

Thanks for my master advisor Professor Pingzhi Fan for leading me to the scientific research and providing opportunities to broaden my eyesight on it.

Acknowledgement is also extended to all members of the Network Laboratory who have once been, or are being my companions during my PhD research. I thank Dr. Haibin Kan and Dr. Xiaohong Jiang for their kind help on both of my research and life in JAIST. It was also a great pleasure to work with Dr. Gui Xie, Dr. Keqiu Li, Dr. Zonghua Zhang, Dr. Hui Tian, Dr. Wenyu Qu, Dr. Chao Peng, Ms. Qing Gong, Mr. Haibo Zhang, Ms. Yawen Chen, Mr. Shihong Xu, Mr. Junya Nakata, Mr. Yosiki Makino, Mr. Junson Kim, Mr. Qinghui Song, Ms. Megumi Aikawa, Mr. Akinori Isogai, Mr. Takashi Okada, Mr. Yingjin Chi, Mr. Yongguang Jin, Mr. Taichi Nakamura, Mr. Ryousuke Sakano, Mr. Kouichirou Torii, and Mr. Takahiro Fukuda.

My special thanks to the Graduate Research Program of JAIST for supporting my research and life for three years.

Finally I devote this thesis to all members of my family for their unconditional love and support, even though they know little about my work.

Contents

Abstract i

Acknowledgments iii

1 Introduction 1
- 1.1 Thesis Statement . . . 1
- 1.2 Our Results . . . 2
- 1.3 Preliminaries . . . 3
  - 1.3.1 The Ideal/Real Model Paradigm . . . 3
  - 1.3.2 Computational Indistinguishability . . . 4
  - 1.3.3 Security in the Semi-Honest Model . . . 4
  - 1.3.4 Security with respect to Malicious Behavior . . . 5
  - 1.3.5 Homomorphic Encryption . . . 7
- 1.4 Organization of the Dissertation . . . 7

2 Secure Two-Party Vector Dominance 9
- 2.1 Problem Background . . . 9
- 2.2 Related Work . . . 10
  - 2.2.1 Related Work from General Solutions . . . 10
  - 2.2.2 Related Work from Millionaire Protocols . . . 10
  - 2.2.3 Related Work from other STVD Protocols . . . 11
- 2.3 Problem Definition . . . 11
- 2.4 Basic Tools . . . 12
  - 2.4.1 Homomorphic Encryption . . . 12
  - 2.4.2 Zero-knowledge Proofs . . . 13
  - 2.4.3 0-encoding . . . 14
- 2.5 Main Idea and Building Blocks . . . 15
  - 2.5.1 Random-zero Transformation . . . 15
  - 2.5.2 Privacy Preserving Product of Scalar Products . . . 16
- 2.6 The Protocol for STVD in the Semi-honest Case . . . 16
- 2.7 The Protocol for STVD in the Malicious Case . . . 18
  - 2.7.1 Security against Malicious Bidder . . . 18
  - 2.7.2 Security against Malicious Seller . . . 18
  - 2.7.3 The Protocol for the malicious case . . . 19
- 2.8 Comparisons with other solutions . . . 22
  - 2.8.1 Complexities of Protocol 1 and Protocol 2 . . . 22
  - 2.8.2 A Derived STVD Solution from [84] . . . 23

2.8.3 Comparisons . . . 24
2.9 Conclusions . . . 24
2.10 Appendix . . . 25
2.10.1 Major Notations . . . 25
2.10.2 Proofs of Lemmas for Building Blocks . . . 25
2.10.3 Some calculations on complexities . . . 27
2.10.4 Some Basic Zero-knowledge Proofs . . . 28

3 Privacy Preserving Set Intersection among Multiple Parties 30
3.1 Introduction . . . 30
3.2 Related Work . . . 31
3.3 Problem Definition . . . 31
3.4 Basic Tools . . . 32
3.4.1 Homomorphic Encryption . . . 32
3.4.2 Calculations on encrypted polynomials . . . 33
3.4.3 Notations . . . 34
3.5 Protocol for Privacy Preserving Set Intersection in the semi-honest model . 34
3.5.1 Main Idea . . . 34
3.5.2 The Protocol . . . 35
3.6 Protocol for Privacy Preserving Set Intersection in the malicious model . . 37
3.6.1 Zero-Knowledge Proofs . . . 37
3.6.2 Protocols for the Malicious Model . . . 37
3.7 Security Analysis . . . 39
3.7.1 Security Analysis on Protocol 1 . . . 39
3.7.2 Security Analysis on Protocol 2 . . . 41
3.8 Comparisons with Related Work . . . 41
3.8.1 Comparisons for Protocol 1 . . . 41
3.8.2 Comparisons for Protocol 2 . . . 42
3.9 Concluding Remarks . . . 43
3.10 Appendix . . . 43
3.10.1 Proofs of Theorems . . . 43
3.10.2 Generation of Random and Nonsingular Matrix . . . 46
3.10.3 Some Basic Zero-knowledge Proofs . . . 47

4 Privacy Preserving Tuple Matching in Distributed Database 49
4.1 Problem Background . . . 49
4.2 Related Work . . . 51
4.2.1 Related Work for PPTM . . . 52
4.2.2 Related Work for PPTAM . . . 52
4.3 Preliminaries . . . 53
4.3.1 Adversary Model . . . 53
4.3.2 Homomorphic Encryption . . . 53
4.3.3 Calculations on encrypted polynomials . . . 53
4.3.4 Notations . . . 54
4.4 Privacy Preserving Tuple Matching . . . 54
4.4.1 Main Idea . . . 54
4.4.2 The Protocol . . . 56

4.5 Privacy Preserving Threshold Attributes Matching . . . 57
4.5.1 Main Idea . . . 57
4.5.2 The Protocol . . . 57
4.6 Security Analysis . . . 60
4.6.1 Security Analysis for Protocol 1 . . . 60
4.6.2 Security Analysis for Protocol 2 . . . 61
4.7 Comparisons with Related Work . . . 63
4.7.1 Comparisons for Protocol 1 . . . 64
4.7.2 Practical Considerations and Comparisons for Protocol 1 . . . 65
4.7.3 Comparison of Protocol 2 with Solution D2 . . . 66
4.7.4 Practical Considerations and Comparisons for Protocol 2 . . . 67
4.8 Concluding Remarks and Open Problems . . . 67
4.9 Appendix . . . 68

5 Privacy State Test in Wireless Sensor Networks 70
5.1 Problem Background . . . 70
5.2 Privacy Issues in Pervasive Sensor Networks . . . 71
5.3 Related Work . . . 73
5.3.1 Key Management Schemes in Wireless Sensor Networks . . . 73
5.3.2 Related Work for Protecting Originator's Location Privacy . . . 75
5.4 Fundamental Problems of Protocols . . . 75
5.4.1 Two Fundamental Problems . . . 75
5.4.2 Secure Two-party Point-Inclusion Protocol . . . 77
5.5 The Scheme to Test Privacy State . . . 78
5.5.1 Who's Alice . . . 78
5.5.2 Who's Bob . . . 78
5.5.3 The Architecture . . . 78
5.5.4 The Scheme . . . 78
5.6 Evaluation of Our Scheme . . . 79
5.7 Chapter Summary and Future Work . . . 79

6 Conclusions and Future Work 81
6.1 Contributions . . . 81
6.2 Future Research Directions . . . 83

References 85

Publications 92

Chapter 1

Introduction

1.1 Thesis Statement

In distributed scenarios, parties usually need do computation tasks over their inputs. Due to their privacy or secrecy, the parties may dislike to publish the inputs to each other. If there exists a party which can be trusted by all involved parties, they can send their inputs only to the trusted party, then the trusted party is in charge of the computation, and things become much easy. However, in many cases the assumption of a trusted party may be impractical. For example, an alliance of supermarkets need know the credit card numbers of customers who have purchasing records in all of their databases, and provide discounts to those customers. It is hard for the supermarkets to find a really "trustful" party to which they would send their customers database, and even if they find such service sanctioned by the government, they may think it uneconomical because they need pay high for the service. What's more, there may be some dishonest parties who will do attacks in the computation, e.g., analyzing or arbitrarily substituting the intermediate computation, or arbitrarily quitting the computation whenever it gets desired results. Therefore, how to protect the privacy of the honest parties's inputs against the attacks of the dishonest parties, without the help of a trusted party, is a critical problem required by many practical applications.

Thus, the terminology *Secure Multi-party Computation* (SMC) has been put forward to deal with computing any probabilistic function on any input, in a distributed network where each participant holds one of the inputs, ensuring independence of the inputs, correctness of the computation, and that no more information is revealed to a participant in the computation than can be computed from that participant's input and output ([50]).

There have been general solutions for the SMC problem ([49], [92]). In general SMC, the function to be computed is represented by a circuit, and every gate of the circuit is privately evaluated. While this solution (circuit evaluation) is appealing in its generality and simplicity, the complexity of the protocol it generates depends on the size of the circuit. This size depends on the size of the input and on the complexity of expressing the function as a circuit. As pointed out in [48], this general solution is impractical for a specific problem, because the large size of the circuit will result in a much less efficient protocol than the non-private protocol for this problem.

For some specific problems, their properties have been utilized to produce more efficient private protocols without employing the general circuit evaluation solution. In this paper, we will focus on a few specific SMC problems in e-bidding, database, and wireless sensor

networks, follow the idea of utilizing properties of the specific problems, and aim at more efficient solutions than related work without employing the solution of circuit evaluation in general SMC.

1.2 Our Results

Specifically, our problems and results include the following:

1) **Secure Two – party Vector Dominance** : In the multi-commodity private bidding, a manufacturer may only want to deal with the supplier that can simultaneously satisfy the requirement for N items. This is a *Secure Two-party Vector Dominance* (STVD) problem, in which Alice needs to learn whether each coordinate of her private vector is larger than the corresponding coordinate of Bob's private vector. We formally define the problem, prove the completeness and soundness of our proposed protocols, and their security under the semi-honest and malicious models. By parallel executions, our protocol for the semi-honest case need less modular exponentiations than previous results from [84] and [4], and our protocol for the malicious case has the same level of computation with the derived solution from [84].

2) **Privacy Preserving Set Intersection** : In a distributed scenario, N parties need to learn the intersection of their private data sets, and each party cannot know more elements on the other parties than the intersection. This is the *Privacy Preserving Set Intersection* (PPSI) problem. In comparison with related work, our protocol for PPSI is more efficient in computation and communication costs while keeping the security both in the semi-honest and malicious model.

3) **Privacy Preserving Tuple Matching** and **Privacy Preserving Threshold Attributes Matching** : In a scenario of horizontally distributed databases, each party needs to learn the fact that whether a tuple inside its database can be matched in any private database of the other parties. At the same time, the party should not learn the number of the matched tuples and the identities of the matched parties. This is the *Privacy Preserving Tuple Matching* (PPTM) problem. Compared with related work, our protocol for PPTM is more efficient in computation, need an increased communication cost which can be transferred within a few seconds, while keeping the security in the semi-honest model. We also solve another relevant problem, *Privacy Preserving Threshold Attributes Matching* (PPTAM), in which each party wants to learn that whether each attribute of its tuple appears at least 2 times in the attribute union of the N parties. To our knowledge we are the first to tackle the PPTAM problem. Though the techniques in [62] can provide a derived solution for PPTAM, our protocol is more efficient in computation and communication costs in comparisons.

4) **Location Privacy Protection in Pervasive Sensor Networks** : In the circumstances of pervasively deployed sensor networks, people will have a dynamic concern about their location privacy. At the same time, sensors will become invisible or should be hidden due to the privacy of themselves. To our knowledge we are the first to discuss dynamic location privacy issues in pervasively deployed sensor

networks and propose a scheme for people in the environment of pervasive sensor networks, so that they can be aware of whether they should be alert on their privacy activities. Based on the protocol of secure two-party point inclusion in which our STVD protocol is used as a building block, our scheme has the characteristics of generality and confidentiality.

1.3 Preliminaries

1.3.1 The Ideal/Real Model Paradigm

The aim of an SMC task is for the participating parties to *securely* and correctly compute some function of their distributed and private inputs. The security properties that should be preserved are various in different applications. In some applications, each participant should learn nothing about others' inputs, but in some applications, the output of the computation may reveal some information on others' inputs. (e.g., in the problem of privacy preserving set intersection, the inputs in the intersection set are revealed to all participants.) Another important property is correctness, which means the parties' output is really that defined by the function, whether or not the parties are honest. Therefore, a general definition covering the correctness property and all the application-dependent security properties is a basic requirement in the study of SMC. Till now, the dominant paradigm is the *ideal/real model paradigm*, a formal definition of which can be found in [12].

Loosely speaking, this paradigm defines the security of a real protocol by comparing it to an *ideal computing scenario* in which the parties interact with an external trusted and incorruptible party. In this ideal execution, the parties all send their inputs to the trusted party (via ideally secure communication lines). The trusted party then computes the function on these inputs and sends each party its specified output. Such a computation embodies the goal of secure computation. Specifically, it is clear that the privacy property as defined above holds, because the only message that a party receives (even one who behaves maliciously) is the output. Likewise, since the trusted party is incorruptible, correctness is also guaranteed to hold. In addition to the fact that the above and other properties are preserved in an ideal execution, the simplicity of the ideal model provides an intuitively convincing security guarantee. For example, notice that the only message that a party sends in an ideal execution is its input, and so the only power that a corrupted party has is to choose its input (something which is typically legitimate anyway).

So far, we have defined an ideal execution in an ideal world. However, in the *real world*, there is no external party that can be trusted by all parties. Rather, the parties run some protocol amongst themselves without any outside help. Nevertheless, we would like a real protocol to somehow "emulate" an ideal execution. That is, we say that a real protocol that is run by the parties (in a world where no trusted party exists) is secure, if no adversary can do more harm in a real execution than in an execution that takes place in the ideal world. Stated differently, for any adversary carrying out a successful attack on a real protocol, there exists an adversary that successfully carries out the same attack in the ideal world. This suffices because, as we have seen, no successful attacks can be carried out in an ideal execution. Thus, no successful attacks can be carried out on the real protocol, implying that it is secure.

1.3.2 Computational Indistinguishability

Generally speaking there are two types of adversaries in SMC, depending on whether they take active steps to disrupt the execution of the protocol, or merely gather information. The latter adversary is referred to as *semi-honest* (or *passive*, *honest-but-curious*); the former one is referred to as *malicious* (or *active*). In SMC, the adversaries are assumed to be *probabilistic polynomial-time* (PPT) bounded. The security in both types are argued by the *computational indistinguishability* of the views in the ideal model and real model ([48, 66]).

The concept of computational indistinguishability is a kind of equivalence between distributions based on the concept of efficient computation: Distributions are considered to be computationally equivalent if they cannot be differentiated by any efficient algorithm. Computational indistinguishability is also called *indistinguishability in polynomial time* ([47]). Suppose an ensemble $X = \{X_n\}_{n\in N}$ is a sequence of random variables X_n for $n \in N$, which are ranging over strings of length $poly(n)$. Two ensembles $X = \{X_n\}_{n\in N}$ and $Y = \{Y_n\}_{n\in N}$ are indistinguishable in polynomial time, denoted by "$X \equiv^c Y$", if for every PPT algorithm A, and every $c > 0$, there exists an integer N such that for all $n \geq N$,

$$|Pr[A(X_n) = 1] - Pr[A(Y_n) = 1]| < \frac{1}{n^c},$$

where $Pr[A(x) = 1]$ is the probability that A outputs 1 on input x.

1.3.3 Security in the Semi-Honest Model

A semi-honest party is one who follows the protocol properly, with the exception that it keeps a record of all its intermediate computations. Semi-honest parties do constitute a model of independent interest ([48]). Because deviation from the specified program is difficult in many settings, general malicious behavior, e.g., aborting the protocol or entering the protocol with an input other than the one provided, may be infeasible for many users. However, it is easier to record the contents of some registers by the standard activities of the operating system, so some semi-honest behaviors may be feasible for the users: they may analyze their records of all intermediate computations so as to get any information other than the output.

Security in this model means that no player or coalition of players gains information which is not inherent in the output of the calculated function. In the following, we will give the formal definitions on security in the semi-honest model respectively in the two-party and multi-party cases. All the definitions are from [48], and will be employed to formally prove the security of our protocols in the following chapters.

Two-Party Case

Definition 1 *(Security in the semi-honest model, two-party case) Let* $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ *be a functionality, that is,* $f(x,y) = (f_1(x,y), f_2(x,y))$ *where* $f_1(x,y)$ *and* $f_2(x,y)$ *are desired outputs of two parties respectively. Let* Π *be a two-party protocol for computing* f*, and* $\Pi_1(x,y)$*,* $\Pi_2(x,y)$ *are outputs of two parties respectively.*

Let $\overline{\mathcal{M}} = (\mathcal{M}_1, \mathcal{M}_2)$ *be a pair of PPT algorithms representing the two parties' strategies respectively for computing* f *in the* ideal model. $\overline{\mathcal{M}}$ *is said to be* **admissible** *if at least one* $\mathcal{M}_k$ $(k \in \{1,2\})$ *is honest, i.e., following the ideal execution. The*

joint execution of f under $\overline{\mathcal{M}}$ in the ideal model, *denoted* $IDEAL_{f,\overline{\mathcal{M}}}(x,y)$, *is defined as the output pair of* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *in the ideal execution.*

Let $\overline{M} = (M_1, M_2)$ *be a pair of PPT algorithms representing the two parties' strategies respectively for executing* Π *in the* real model. $\overline{M}$ *is said to be* **admissible** *if at least one* M_k $(k \in \{1,2\})$ *is honest. The* **joint execution of Π under $\overline{\mathbf{M}}$ in the real model**, *denoted* $REAL_{\Pi,\overline{M}}(x,y)$, *is defined as output pair of* M_1 *and* M_2 *resulting from the protocol interaction.*

In case $\mathcal{M}_1$ *in the ideal model is semi-honest,* $IDEAL_{f,\overline{\mathcal{M}}}(x,y) = (\mathcal{M}_1(x, f_1(x,y)), f_2(x,y))$. *In case* M_1 *in the real model is semi-honest,* $REAL_{f,\overline{M}}(x,y) = (M_1(x, \Pi_1(x,y)), \Pi_2(x,y))$. $\mathcal{M}_1(x, f_1(x,y))$ *and* $M_1(x, \Pi_1(x,y))$ *are views of* $\mathcal{M}_1$ *and* M_1 *on their input, output, randomness and public transcript in the executions.*

Protocol Π *is said to* **securely compute f in the semi − honest model** *if*

$$\{IDEAL_{f,\overline{\mathcal{M}}}(x,y)\}_{x,y,s.t.,|x|=|y|} \equiv^c \{REAL_{\Pi,\overline{M}}(x,y)\}_{x,y,s.t.,|x|=|y|}.$$

In this definition, the views of semi-honest adversaries in the ideal and real models are included in the joint executions of f and Π respectively. For example, if $\mathcal{M}_1$ in the ideal model is semi-honest, , the part of $\mathcal{M}_1(x, f_1(x,y))$ in $IDEAL_{f,\overline{\mathcal{M}}}(x,y)$ denotes view of $\mathcal{M}_1$ on its input, output, randomness and public transcript in the executions. If M_1 in the real model is semi-honest, the part of $M_1(x, \Pi_1(x,y))$ in $REAL_{f,\overline{M}}(x,y)$ denotes view of M_1 on its input, output, randomness and public transcript in the executions.

Multi-Party Case

Definition 2 *(privacy with respect to semi-honest behavior, multi-party case) Let* $f : (\{0,1\}^*)^m \to (\{0,1\}^*)^m$ *be an m-ary functionality, where* $f_i(x_1, ..., x_m)$ *is the i-th element of* $f(x_1, ..., x_m)$. *For* $I = \{i_1, ..., i_c\} \subseteq \{1, ..., m\}$, $f_I(x_1, ..., x_m) = \{f_{i_1}(x_1, ..., x_m), ..., f_{i_c}(x_1, ..., x_m)\}$. *Let* Π *be an m-party protocol for computing* f. *The* **view** *of the i-th party* (P_i) *after participating in an execution of* Π *on* $\overline{x} = (x_1, ..., x_m)$, *denoted* $VIEW_i^{\Pi}(\overline{x})$, *is* $(x_i, r, m_1, ..., m_t)$, *where* r *are the random bits generated by* P_i, $m_1, ..., m_t$ *is a sequence of message received by* P_i. *For* $I = \{i_1, ..., i_c\}$, *we let* $VIEW_I^{\Pi}(\overline{x}) = (I, VIEW_{i_1}^{\Pi}(\overline{x}), ..., VIEW_{i_c}^{\Pi}(\overline{x}))$.

We say that Π **Privately Computes** f *if there exists a probabilistic polynomial-time (PPT) algorithm, denoted* S, *such that for every* $I \subseteq \{1, ..., m\}$, *it holds that*

$$S(I, (x_{i_1}, ..., x_{i_c}), f_I(\overline{x}))_{\overline{x} \in (\{0,1\}^*)^m} \equiv^c VIEW_I^{\Pi}(\overline{x})_{\overline{x} \in (\{0,1\}^*)^m}$$

Suppose the index set of semi-honest parties is denoted by one $I = \{i_1, ..., i_c\} \subseteq \{1, ..., m\}$, then in Definition 2, $S(I, (x_{i_1}, ..., x_{i_c}), f_I(\overline{x}))$ denotes view of a semi-honest adversary in the ideal model. $VIEW_I^{\Pi}(\overline{x})$ denotes view of a semi-honest adversary in the real model.

1.3.4 Security with respect to Malicious Behavior

A *malicious* party may arbitrarily deviate from the specified program of a protocol. Specifically there are three things we cannot hope to avoid in any protocol:

1) Parties refusing to participate in the protocol when the protocol is first invoked.

2) Parties arbitrarily substituting its original local input and entering the protocol with an input other than the one provided to them.

3) Parties aborting the protocol whenever obtaining the desired result.

In [48], it has been proved that any protocol for the semi-honest model can be compiled into an "equivalent" protocol for the malicious model. In this thesis we will fix the protocols for the semi-honest model to be secure in the malicious model, following the ideas of the compiler in [48].

We will not consider *perfect fairness* in our protocols. As pointed out in [48], perfect fairness - in the sense of both parties obtaining the outcome of the computation concurrently - is not achievable in a two-party computation, because a malicious party may abort the protocol permaturely before sending the last message. In the case of multi-party computation, when the number of parties that deviate from the protocol is arbitrary, then perfect fairness cannot also be achieved.

In the following we give the formal definitions of security in the malicious model for the two-party computation and multi-party computation respectively. In the definition of the security in the multi-party computation, we assume that the number of parties that deviate from the protocol is arbitrary. The definitions will be used to define our problems and prove the security of our protocols in the following chapters.

Two-Party Case

Definition 3 *(Security in the malicious model, two-party case) Let* $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \times \{0,1\}^*$ *be a functionality, that is,* $f(x,y) = (f_1(x,y), f_2(x,y))$ *where* $f_1(x,y)$ *and* $f_2(x,y)$ *are desired outputs of two parties respectively. Let* $\overline{\mathcal{M}} = (\mathcal{M}_1, \mathcal{M}_2)$ *be a pair of PPT algorithms representing the two parties' strategies respectively for computing* f *in the* ideal model. $\overline{\mathcal{M}}$ *is said to be* **admissible** *if at least one* $\mathcal{M}_k$ $(k \in \{1,2\})$ *is honest, i.e., following the ideal execution. The* **joint execution of f under** $\overline{\mathcal{M}}$ **in the ideal model**, *denoted* $IDEAL_{f,\overline{\mathcal{M}}}(x,y)$, *is defined as the output pair of* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *in the ideal execution.*

Suppose in the ideal model the trusted party firstly sends $f_1(x,y)$ *to* $\mathcal{M}_1$, *and then sends* $f_2(x,y)$ *to* $\mathcal{M}_2$. *In case that* $\mathcal{M}_1$ *is malicious and always aborts,* $IDEAL_{f,\overline{\mathcal{M}}}(x,y) = (\mathcal{M}_1(x,\perp),\perp)$. *In case that* $\mathcal{M}_1$ *never aborts,* $IDEAL_{f,\overline{\mathcal{M}}}(x,y) = (\mathcal{M}_1(x, f_1(x',y)), f_2(x',y))$ *where* $x' = \mathcal{M}_1(x)$ *is the input* $\mathcal{M}_1$ *gives to the trusted party.*

Let Π *be a two-party protocol for computing* f, *and* $\Pi_1(x,y)$, $\Pi_2(x,y)$ *are outputs of two parties respectively. Let* $\overline{M} = (M_1, M_2)$ *be a pair of PPT algorithms representing the two parties' strategies respectively for executing* Π *in the* real model. $\overline{M}$ *is said to be* **admissible** *if at least one* M_k $(k \in \{1,2\})$ *is honest. The* **joint execution of** Π **under** $\overline{\mathbf{M}}$ **in the real model**, *denoted* $REAL_{\Pi,\overline{M}}(x,y)$, *is defined as output pair of* M_1 *and* M_2 *resulting from the protocol interaction.*

Protocol Π *is said to* **securely compute f in the malicious model** *if*

$$\{IDEAL_{f,\overline{\mathcal{M}}}(x,y)\}_{x,y,s.t.,|x|=|y|} \equiv^c \{REAL_{\Pi,\overline{M}}(x,y)\}_{x,y,s.t.,|x|=|y|}.$$

In the above definition, the symbol "$\perp$" means aborting of a party.

Multi-Party Case

Definition 4 *(Security in the malicious model, multi-party case) Let* $f : (\{0,1\}^*)^m \rightarrow (\{0,1\}^*)^m$ *be an* m*-ary functionality, where* $f_i(x_1,...,x_m)$ *is the* i*-th element of* $f(x_1,...,x_m)$. *Let* $I = \{i_1,...,i_c\} \subseteq \{1,...,m\}$, *and* $\overline{I} = \{1,...,m\} \setminus I$. $f_I(x_1,...,x_m) = \{f_{i_1}(x_1,...,x_m),...,$

$f_{i_c}(x_1, ..., x_m)\}$. *A pair* $(I, \mathcal{M})$*, where* $\mathcal{M}$ *is a PPT algorithm, represents an adversary in the ideal model. The* **joint execution of f under** $(\mathbf{I}, \mathcal{M})$ **in the ideal model** *is denoted by* $IDEAL_{f,I,\mathcal{M}}(\overline{x})$.

Suppose in the ideal model the trusted party firstly sends $f_1(\overline{x}')$ *to Party 1. In case that Party 1 is malicious and always aborts,* $IDEAL_{f,I,\mathcal{M}}(\overline{x}) = (\mathcal{M}(\overline{x}_I, f_I(\overline{x}'), \perp)$. *In case that Party 1 is malicious and never aborts,* $IDEAL_{f,I,\mathcal{M}}(\overline{x}) = (\mathcal{M}(\overline{x}_I, f_I(\overline{x}')), f_{\overline{I}}(\overline{x}'))$*, where* $\overline{x}' = (x'_1, ..., x'_m)$ *such that* $x'_i = \mathcal{M}(\overline{x}_I)$ *for* $i \in I$ *and* $x'_i = x_i$ *otherwise.*

Let Π *be an m-party protocol for computing f. A pair* (I, M)*, where M is a PPT algorithm, represents an adversary in the real model. The* **joint execution of Π under** $(\mathbf{I}, \mathbf{M})$ **in the real model** *is denoted by* $REAL_{\Pi,I,M}(\overline{x})$*, is defined as the output sequence resulting from the interaction between the m parties, where the messages of parties in I are computed according to* $M(\overline{x}_I)$ *and the messages of parties in* $\overline{I}$ *are computed according to* Π.

Protocol Π *is said to* **securely compute f in the malicious model** *if*

$$\{IDEAL_{f,I,\mathcal{M}}(\overline{x})\}_{\overline{x}} \equiv^c \{REAL_{\Pi,I,M}(\overline{x})\}_{\overline{x}}.$$

1.3.5 Homomorphic Encryption

A few of our protocols are based on an additive Homomorphic Encryption (HE) scheme. Let ε be a probabilistic encryption scheme. Let M be the message space and C the ciphertext space such that M is a group under operation $\boxplus$ and C is a group under operation $\boxdot$. ε is a $(\boxplus, \boxdot)$-HE scheme if for any instance $E_R(\cdot)$ of the encryption scheme, given $c_1 = E_{r_1}(m_1)$ and $c_2 = E_{r_2}(m_2)$, there exists an r such that

$$c_1 \boxdot c_2 = E_{r_1}(m_1) \boxdot E_{r_2}(m_2) = E_r(m_1 \boxplus m_2)$$

ε is *additively homomorphic* when it is a $(+, \boxdot)$ scheme, and *multiplicatively homomorphic* when it is a $(*, \boxdot)$ scheme.

The HE scheme in our protocols is also required to support secure (N, N)-threshold decryption. The corresponding secret key is shared by a group of N parties, and the decryption cannot be performed by any single party, unless all parties act together.

Thus, the candidates of threshold HE schemes include ElGamal ([25]) and Paillier cryptosystems ([75]). Both of them have the following properties:

1) they are additive homomorphic encryption schemes. Given two encryptions $E(m_1)$ and $E(m_2)$, $E(m_1 + m_2) = E(m_1) \cdot E(m_2)$;

2) given an encryption $E(m)$ and a scalar a, $E(a \cdot m) = E(m)^a$;

3) (N, N)-threshold decryption can be supported (by [44], [84], [33],[34]).

1.4 Organization of the Dissertation

In Chapter 2 we will discuss the problem of STVD. We will define the problem firstly. Then we will discuss some related work, basic tools, the main idea and building blocks. Our protocols for semi-honest case and malicious case will be described and analyzed respectively. Finally we will compare our work with previous work.

In Chapter 3 we will define the problem of PPSI, and discuss some related work on PPSI. The PPSI protocol for the semi-honest and malicious model will be proposed respectively, and their security will be analyzed. Finally our protocols will be compared with the related work considering the computation and communication costs.

In Chapter 4 we will address the PPTM and PPTAM problems. Some related work and necessary preliminaries for our protocols will be listed. The PPTM and PPTAM protocols will be proposed respectively. Their correctness will be proved and their security will be analyzed. We will compare our protocols with the related work considering the computation and communication costs, and get their response times by experiments in a moderate-scale application.

In Chapter 5 we will address the issues on testing privacy state in pervasive sensor networks. Terminologies about sensing area privacy and originator privacy will be discussed. Some related work will be outlined. The definition, building blocks, protocol about the secure two-party point-inclusion problem will be described. Then the specific scheme to test privacy state in pervasive sensor networks will be presented.

Chapter 2

Secure Two-Party Vector Dominance

2.1 Problem Background

Let $A = (a_1, a_2, ..., a_n)$ and $B = (b_1, b_2, ..., b_n)$ be two vectors. a_i and b_i $(i = 1, ..., n)$ are K-bit integers. We say A *dominates* B (denoted by $A \succeq B$), if $a_i \geq b_i$ for all $i = 1, ..., n$. If there is at least one ordering $a_j < b_j$ $(1 \leq j \leq n)$, A does not dominate B (denoted by $A \not\succeq B$). In multi-commodity bidding, a seller may only want to deal with the bidder who can simultaneously satisfy the requirements for n commodities because, e.g., they have some coordinations in the production. This can be treated as a *vector dominance* problem: the bidder has a bidding vector A, the seller has a sale price vector B, and they will make a deal if $A \succeq B$.

Privacy may be a great concern for both the seller and bidder. Due to commercial secrecy, whether $A \succeq B$ or not, one party should not know any specific element in the other party's vector. If $A \not\succeq B$, both of them know there is at least one pair (a_j, b_j) in which $a_j < b_j$, but neither of them should know the specific number of these pairs, and the ordering of any specific pair (a_i, b_i) (i.e., whether $a_i < b_i$ or not). In this chapter, we address the problem of *secure two-party vector dominance* (STVD), i.e., determining whether $A \succeq B$ without disclosing any information other than the determination to the two parties.

STVD is a multi-dimensional extension of Yao's millionaire problem [92], in which two parties respectively holding value a and b determine which one is greater without disclosing a and b to each other. However, STVD can not be solved by n trivial executions of a protocol for the millionaire problem, because the orderings of element pairs should be kept private in the case of no dominance.

Our main contributions in this chapter include:

1) We define the problem of STVD in terms of completeness, soundness and security in both semi-honest model and malicious model.

2) We propose an STVD protocol which is proved to be overwhelmingly complete and sound, and be secure in the semi-honest model. Given K is the length of each element in the vector, in $K+1$ parallel execution in which Alice has K platforms and Bob has one platform, our protocol has higher efficiency compared with a derived solution from [84] and another solution from [4].

3) We fix our protocol to be secure against malicious behaviors in multi-commodity

private bidding, and the fixed protocol is also proved to be overwhelmingly complete and sound. In $K+1$ parallel execution, our protocol has the same level of efficiency compared with the derived solution from [84].

The remaining part of this chapter is organized as follows. Some related work is discussed in Section 2.2. The problem of STVD is defined in Section 2.3. Some basic tools used in our protocols are given in Section 2.4. The main idea and building blocks are described in Section 2.5. Our protocols for semi-honest case and malicious case are described and analyzed respectively in Section 2.6 and 2.7. Section 2.8 compares our work with previous work. The chapter is concluded in Section 2.9. In the appendix we illustrate some major notations of this chapter and give detailed proofs on some lemmas.

2.2 Related Work

2.2.1 Related Work from General Solutions

STVD is a specific problem belonging to the general *Secure Multiparty Computation* (SMC) problem. There have been general solutions for the SMC problem ([48, 92]). In general SMC, the function to be computed is represented by a circuit, and every gate of the circuit is privately evaluated. However, when this general solution is used for a specific problem, the large size of the circuit and high cost of evaluating each gate will result in a much less efficient protocol than the non-private protocol for this problem. Therefore, many efficient private protocols for the specific problems have been proposed without using too many complex circuits and evaluation gates.

2.2.2 Related Work from Millionaire Protocols

Millionaire problem has been extensively studied ([6, 10, 32, 57, 64, 65, 76, 84, 92]). However, it's unsuitable to trivially run these protocols on every pair (a_i, b_i) for $i = 1, ..., n$ to check whether $A \succeq B$, otherwise the ordering of every pair will be leaked. A possible solution is to use a homomorphic encryption scheme to joint the comparison result of every pair from a millionaire protocol and determine the dominance by decrypting the final joint result. However, to achieve this solution by millionaire protocols in [32, 65, 76], an encryption scheme which supports both additive and multiplicative homomorphism, or both AND and XOR homomorphism, should be required, which is an open difficulty till now.

Both [64] and [84] have also provided circuits to privately compare two values a and b. In the two circuits, the output t is 1 if $a > b$, and 0 otherwise. In [84] *conditional gate* is also provided for AND operation on two encryptions, i.e., computing $E(xy)$ given $E(x)$, $E(y)$, $x \in \{-1, 1\}$, $y \in \mathbb{Z}_q$, and an additively homomorphic encryption scheme $E(\cdot)$. Then a derived solution for STVD based on their work can be two steps: 1) compute the output $E(t_i)$ by putting a_i and b_i into the Millionaire circuit for $i = 1, ..., n$, 2) compute $E(\prod_{i=1}^{n} t_i)$ by $n-1$ conditional gates. If $\prod_{i=1}^{n} t_i = 1$, $A \succ B$; if $\prod_{i=1}^{n} t_i = 0$, $A \not\succ B$. Notice that $E(t_{i-1}t_i)$ can not be computed directly by conditional gate because t_{i-1} and $t_i \in \{0, 1\}$. So $E(t'_i) = E(2t_i - 1)$ can be firstly computed in which $t'_i = 2t_i - 1 \in \{-1, 1\}$, then $E(t_{i-1}t'_i)$ is computed by conditional gate, and $E(t_{i-1}t_i) = E(\frac{t_{i-1}t'_i + t_{i-1}}{2})$. In Section 2.8 we will compare this derived solution with our STVD protocol.

2.2.3 Related Work from other STVD Protocols

There have been a few protocols for STVD which were all constructed on the semi-honest model. The protocol proposed in [55] may leak some sensitive information. For example, to compare two vectors (a_1, a_2) and (b_1, b_2), in [55] the two parties privately computed $S = (a_1 - b_1)(a_2 - b_2) - |(a_1 - b_1)(a_2 - b_2)|$. When the two vectors have a dominance relation, $S = 0$, but when they have no dominance relation, the value of S will help a semi-honest party to guess the other's vector. The protocol in [4] disguised the two vectors by extending each n-dimension vector to be a $4n$-dimension vector, then used three building blocks: private permutation, millionaire protocol, and private equality-test, to determine the dominance. Though each building block can be replaced by the most efficient protocol proposed for the corresponding problem, they are always based on different encryption schemes. At the end of each block, the intermediate computation need be decrypted, and at the beginning of another block, the computation need be encrypted again. Therefore it's difficult to prevent a malicious party from arbitrarily substituting the intermediate computation. We will also compare this solution from [4] with our STVD protocol in Section 2.8.

In [81], based on the technique of 0-encoding in [65], we had employed a building block to avoid disclosing the ordering of each pair (a_i, b_i), and STVD had been solved by an encryption scheme which is only additively homomorphic. The major difference of this chapter from [65] and [81] is that: 1) we change the size of the 0-encoding list of each b_i to the length of b_i. In [65] and [81], this size is rightly the number of bit "1" in b_i. If a malicious party knows this number, he gleans sensitive information on b_i. Our change hides this number and also finds a new lemma (Lemma 3) to prevent the malicious behaviors of the owner of b_i; 2) we use random-zero transformation on a_i and b_i, and transfer STVD to be a problem of sum on product of scalar products. After the transfer, efficient zero-knowledge proofs based on statements about discrete logarithms can be easily utilized to prevent malicious behaviors. Both solutions in [65] and [81] were only for semi-honest model; 3) we give detailed proofs on the correctness and security of the proposed protocols, while in [65] and [81] the correctness and security were argued only in sketch.

2.3 Problem Definition

Let K be the security parameter, that is, the lengths of the two parties' inputs, $|a_i|$ and $|b_i|$, are K for $i = 1, ..., n$. In multi-commodity bidding, 2^K is the number of possible prices on a commodity. The two parties can negotiate different scales on the commodities, e.g., a price of "11" for a commodity with scale "100$" means an actual price of "1100$". Practically the seller will not sell the commodity at a much lower price, and the bidder will not bid it at a much higher price, so K can be set as 5 to 10.

Let λ be an error probability parameter, and $negl(\lambda)$ is a *negligible function* such that for all polynomials $p(\cdot)$ and all sufficiently large $\lambda \in \mathbb{N}$, $negl(\lambda) < 1/p(\lambda)$. In this chapter, λ is the length of prime p in ElGamal encryption, and is sufficient to be set as 1024 in practice.

Definition 5 (**Secure Two – party Vector Dominance**) *Let Alice be the owner of vector $A = (a_1, ..., a_n)$, and Bob be the owner of vector $B = (b_1, ..., b_n)$. For $i = 1, ..., n$,*

$a_i, b_i \in \{0,1\}^K$ and $a_i \neq 0$. Let $f : \{0,1\}^{nK} \times \{0,1\}^{nK} \to \{0,1\} \times \{0,1\}$ be a functionality for Alice and Bob, that is, $f(A,B) = (f_1(A,B), f_2(A,B))$ where $f_1(A,B)$ and $f_2(A,B)$ are outputs of Alice and Bob respectively. $f(A,B) = (1,1)$ if $A \succeq B$, and $f(A,B) = (0,0)$ if $A \not\succeq B$. The problem of Secure Two-party Vector Dominance (STVD) is to find out a protocol Π to compute f, with $\overline{M} = (M_1, M_2)$ be a pair of PPT algorithms representing Alice and Bob's strategies respectively and satisfying the following conditions:

1) ***Admissible**: At least one M_k ($k \in \{1,2\}$) is honest, i.e., following the execution prescribed by Π. At most one M_k is controlled by a PPT bounded adversary.*

2) ***Completeness**: For an honest strategy M_k ($k \in \{1,2\}$), if $A \succeq B$, the probability of M_k outputting 0 is negligible, i.e., $Pr[M_k = 0] \leq negl(\lambda)$.*

3) ***Soundness**: For an honest strategy M_k, if $A \not\succeq B$, the probability of M_k outputting 1 is negligible, i.e., $Pr[M_k = 1] \leq negl(\lambda)$.*

4) ***Security**: $\{IDEAL_{f,\overline{\mathcal{M}}}(A,B)\}_{|A|=|B|=nK} \equiv^c \{REAL_{\Pi,\overline{M}}(A,B)\}_{|A|=|B|=nK}$. $\overline{\mathcal{M}} = (\mathcal{M}_1, \mathcal{M}_2)$ is a pair of PPT strategies respectively for the two parties when computing f in the ideal model.*

Actually the security in the definition has implied the completeness and soundness, i.e., the view of an honest M_k ($k \in \{1,2\}$) in the real model should be computationally indistinguishable with the view of an honest $\mathcal{M}_k$ in the ideal model. We specify the completeness and soundness for clarity and simplifying the security analysis.

In the definition, we also require that $a_i \neq 0$ for $i = 1, ..., n$, in accordance with the multi-commodity private bidding, in which the seller may set some $b_j = 0$ but would not like the bidder to win that item free of charge, so all bids of the bidder are required to be nonzero.

2.4 Basic Tools

2.4.1 Homomorphic Encryption

Our construction of STVD protocol is based on an additive Homomorphic Encryption (HE) scheme. The HE scheme is also required to support secure $(2,2)$-threshold decryption, in which the corresponding secret key is shared by two parties, and the decryption can not be performed by any single party, but only by both parties acting together.

There are a few cryptosystems that satisfy our requirements, and for efficiency we employ ElGamal encryption, the security of which is based on the Decision Diffie-Hellman (DDH) problem. The key of ElGamal encryption can be generated by the secure distributed key generation protocol in [44]. In this chapter we utilize a practical way from [84] to generate the distributed key.

- *Distributed Key Generation*: Let p and q be large primes such that $p = 2q + 1$, and the length of p be λ. G_q denotes $\mathbb{Z}_p^*$'s unique multiplicative subgroup of order q, and G_q has a generator of g. Alice's share of secret key is a random $s_1 \in \mathbb{Z}_q$ and Bob's share of secret key is a random $s_2 \in \mathbb{Z}_q$. Alice and Bob publish $h_1 = g^{s_1}$ and $h_2 = g^{s_2}$, with a proof of knowledge on s_1 and s_2 respectively. The common public key is (g, h) where $h = h_1 h_2 = g^{s_1+s_2}$.

- *Encryption*: given a message $m \in \mathbb{Z}_q$, $E(m, \alpha) = (x, y) = (g^\alpha \ mod \ p, h^\alpha g^m \ mod \ p)$ in which $\alpha \in_R \mathbb{Z}_q$. In this chapter, by "$\alpha \in_R \mathbb{Z}_q$" we mean α is uniformly chosen from $\mathbb{Z}_q$. "$E(m, \alpha)$" is also denoted as "$E(m)$" for simplification.

- *Decryption*: given a ciphertext $c = (x, y)$, Alice and Bob publishes $d_1 = x^{s_1}$ and $d_2 = x^{s_2}$, with a proof that $log_x d_1 = log_g h_1$ and $log_x d_2 = log_g h_2$ respectively. Then $D(c) = y/(d_1 d_2)(mod \ p)$. It should be pointed out that $D(c)$ can not get the plaintext m directly, but only g^m. However, in this chapter we only need know whether $m = 0$, i.e., whether $g^m = 1$.

The scheme has the following properties: 1) given two encryptions $E(m_1) = (x_1, y_1)$ and $E(m_2) = (x_2, y_2)$, we use "$E(m_1) \odot E(m_2)$" to denote "$(x_1 x_2, y_1 y_2)$", then $E(m_1) \odot E(m_2) = E(m_1 + m_2)$; 2) given an encryption $E(m) = (x, y)$ and a scalar a, we use "$E(m)^a$" to denote "(x^a, y^a)", then $E(m)^a = E(am)$.

2.4.2 Zero-knowledge Proofs

In this chapter we use notations such as $PK\{a, b \mid y = g^a h^b\}$, to denote various zero-knowledge proofs such as "zero-knowledge proof of knowledge of integers a and b such that $y = g^a h^b$". The convention is that the elements listed before the sign "|" denote quantities the knowledge of which is being proved and are in general not known to the verifier. By this notation, a proof-protocol can be described by just pointing out its aim while hiding all details.

Efficiently we can construct the following proofs based on proofs of knowledge on statements about discrete logarithms ([83, 37, 16, 58]). The correctness and security of them have been argued respectively in the related work based on the DDH problem. We also describe them in details in the appendix 2.10.4. Our constructions compose the basic proofs using AND ($\wedge$) operations, the closure of which is also argued in [18]. These proofs are used in our protocol for the malicious model.

1) *Proving that the pair of plaintexts is a random-zero pair*: $PK\{r_0, r_1 | C_0 = E(r_0) \wedge C_1 = E(r_1) \wedge (r_0 r_1 = 0) \wedge (r_0 + r_1 \neq 0)\}$. This is to prove that the pair of (r_0, r_1) is either $(0, r_1)$ with $r_1 \in_R \mathbb{Z}_q^*$, or $(r_0, 0)$ with $r_0 \in_R \mathbb{Z}_q^*$ ($\mathbb{Z}_q^* = \{1, ..., q-1\}$). The proof can be based on the following basic proofs:

   1.1) *Proof of knowledge of ElGamal plaintext*: given an ElGamal encryption $E(m, \alpha) = (x, y) = (g^\alpha, h^\alpha g^m)$, the prover P proves to V that he knows α and m such that $x = g^\alpha$ and $y = h^\alpha g^m$. This proof is from Schnorr's protocol ([83]) for proving knowledge of a discrete log x such that $x = g^\alpha$, and Fujisaki-Okamoto's protocol ([37]) for proving knowledge (α, m) such that $y = h^\alpha g^m$.

   1.2) *Proving correct multiplication and the multiplication is zero*: This proof is from [16]. In this proof, P sends $C_0 = E(r_0, \alpha_0)$, $C_1 = E(r_1, \alpha_1)$, $C_2 = C_0^{r_1} \odot E(0, \alpha_2)$ to the verifier. P proves that he knows r_0, α_0, r_1, α_1, α_2 such that C_2 is an encryption of $r_0 r_1$ in which r_0 and r_1 are plaintexts of C_0 and C_1 respectively. Then V decrypts C_2 in the cooperation of P to check whether $r_0 r_1$ is 0.

   1.3) *Proving the sum is nonzero*: This is used to check whether $r_0 + r_1 \neq 0$, and is based on the protocol for plaintext equality test in [58]. P sends $C_0 = E(r_0, \alpha_0)$, $C_1 = E(r_1, \alpha_1)$, $C_3 = E(r_3, \alpha_3)$ with $r_3 \in_R \mathbb{Z}_q^*$, and $C_4 = (C_0 \odot C_1)^{r_3} \odot E(0, \alpha_4)$

to V. P proves he knows r_0, α_0, r_1, α_1, r_3, α_3 and α_4 such that C_4 is an encryption of $(r_0+r_1)r_3$ in which (r_0+r_1) and r_3 are the plaintexts in $C_0 \odot C_1$ and C_3 respectively. Then V decrypts C_4 in the cooperation of P to check whether $(r_0+r_1)r_3 \neq 0$. If $(r_0+r_1)r_3 \neq 0$, he accepts that $r_0+r_1 \neq 0$ without knowing the specific value of r_0+r_1.

2) *Proof of knowing the exponential*: $PK\{r|c = E(x) \wedge C_0 = c^r \odot E(0) \wedge C_1 = E(r)\}$. $c = E(x)$ in which x is unknown to the prover and verifier without decryption. Actually this proof is an variation of proving correct multiplication in the proof 1.2). P should firstly send $C_0 = c^r \odot E(0, \beta)$, $C_1 = E(r, \gamma)$ to V, then prove he knows r, β and γ such that C_0 is an encryption of xr in which r is the plaintext in C_1.

3) *Proving that the prover computes correct exponentiations and multiplications, and some of the pairs of exponentials are random-zero pairs*: $PK\{r_{10}, r_{11}, ..., r_{K0}, r_{K1}|y = c_{10}^{r_{10}} \odot c_{11}^{r_{11}} \odot \cdots \odot c_{K0}^{r_{K0}} \odot c_{K1}^{r_{K1}} \bigwedge_{i=1}^{J}(r_{i0}r_{i1} = 0) \bigwedge_{i=1}^{J}(r_{i0} + r_{i1} \neq 0)\}$ $(J \leq K)$. $c_{10} = E(x_{10})$, $c_{11} = E(x_{11})$, ..., $c_{K0} = E(x_{K0})$, $c_{K1} = E(x_{K1})$, in which x_{10}, x_{11}, ..., x_{K0}, x_{K1} are unknown to both the prover and verifier without decryptions. For the proof, firstly P sends y, $C_{10} = c_{10}^{r_{10}} \odot E(0)$, $C_{11} = c_{11}^{r_{11}} \odot E(0)$, ..., $C_{K1} = c_{K1}^{r_{K1}} \odot E(0)$ to V, and V checks whether $y = C_{10} \odot C_{11} \odot \cdots \odot C_{K1}$. Then P proves $PK\{r_{i0}, r_{i1}|C_{i0} = c_{i0}^{r_{i0}} \odot E(0) \wedge C_{i1} = c_{i1}^{r_{i1}} \odot E(0) \wedge (r_{i0}r_{i1} = 0) \wedge (r_{i0}+r_{i1} \neq 0)\}$ for $i = 1, ..., J$. This proof can be made by firstly proving that P knows the exponentials r_{i0} and r_{i1} by the proof 2), then proving that (r_{i0}, r_{i1}) is a random-zero pair by the proof 1).

2.4.3 0-encoding

Suppose that $a_{iK}...a_{i1} \in \{0,1\}^K$ is the binary representation for a_i, $b_{iK}...b_{i1} \in \{0,1\}^K$ is the binary representation for b_i $(i = 1, ..., n)$. If $a_i > b_i$, there must be an index k $(1 \leq k \leq K)$, which satisfies $b_{ik} = 0$, $a_{ik} = 1$, and $a_{iK}...a_{i(k+1)} = b_{iK}...b_{i(k+1)}$. If such a b_{ik} is substituted by 1, then $a_{iK}...a_{i(k+1)}a_{ik} = b_{iK}...b_{i(k+1)}1$. This is the main idea of 0-encoding on b_i.

Suppose there are J_i zeros in $b_{iK}...b_{i1}$, the *0-encoding* of b_i is the list $S_{b_i}^0 = \{b[i,j]|j = 1, ..., K\}$ in which

$$b[i,j] = \begin{cases} b_{iK}...b_{i(k+1)}1 & \text{if } b_{ik} = 0 \text{ for } K \geq k \geq 1; \\ b_{iK}...b_{i1} & \text{for } j = J_i + 1, ..., K. \end{cases}$$

$S_{b_i}^0$ maybe a multiset. For example, given $b_i = (0101)_2$, $S_{b_i}^0 = \{1, 011, 0101, 0101\}$. The following lemmas describe the properties about $S_{b_i}^0$.

Lemma 1 *If $a_i = 0$, for any K-bit $b_i \in \{0, ..., 2^K - 1\}$, $S_{b_i}^0$ has no prefix of a_i.*

Proof: If $b_i = 0$, $S_{b_i}^0 = \{1, 01, 001, ...\}$, so $S_{b_i}^0$ has no prefix of $\{0\}^K$. If $b_i > 0$, it is easy to see that $S_{b_i}^0$ can not also have a prefix of $\{0\}^K$. ■

Lemma 2 *Given two K-bit numbers a_i $(a_i \neq 0)$ and b_i, $a_i \geq b_i$ if and only if $S_{b_i}^0$ has a prefix of a_i.*

Proof: If $a_i = b_i \neq 0$, $S^0_{b_i}$ must have at least one element $b_{iK}...b_{i1} = a_{iK}...a_{i1}$. If $a_i > b_i$, there must be a k for which $a_{iK}...a_{i(k+1)} = b_{iK}...b_{i(k+1)}$, $a_{ik} = 1$ and $b_{ik} = 0$, so $b_{iK}...b_{i(k+1)}1$ must be in $S^0_{b_i}$ and $b_{iK}...b_{i(k+1)}1 = a_{iK}...a_{i(k+1)}a_{ik}$. On the contrary, if $S^0_{b_i}$ has one prefix of a_i, and if it is $b_{iK}...b_{i1}$, then $b_i = a_i$. If it is intercepted from a "0" bit in b_i, e.g., $b_{iK}...b_{i(k+1)}1 = a_{iK}...a_{i(k+1)}a_{ik}$, then $b_{iK}...b_{i(k+1)} = a_{iK}...a_{i(k+1)}$, $b_{ik} = 0$, and $a_{ik} = 1$. So $a_i > b_i$. ■

E.g., $a = (0110)_2$, $b = (0101)_2$, then $a \geq b$.

Lemma 3 *Suppose $|b[i,j]|$ is the length of $b[i,j]$ and $|b[i,j]| = L_{ij}$ for $j = 1, ..., K$, then $L_{ij} \geq j$.*

Proof: Let J_i be the number of zero bits in b_i. For $j = 1, ..., J_i$, $b[i,j]$ can be viewed as intercepted from the left j-th zero bit in b_i. This zero bit is the left L_{ij}-th bit in b_i, given $|b[i,j]| = L_{ij}$. Because the left j-th zero bit is not always the left j-th bit in b_i, so it is easy to see that $j \leq L_{ij}$. For $j = J_i + 1, ..., K$, $L_{ij} = K$, so $j \leq L_{ij}$. ■

E.g., if $b_i = (0101)_2$, $|b[i,1]| = |(1)_2| \geq 1$, $|b[i,2]| = |(011)_2| \geq 2$, $|b[i,3]| = |(0101)_2| \geq 3$, and $|b[i,4]| \geq 4$.

2.5 Main Idea and Building Blocks

2.5.1 Random-zero Transformation

By Lemma 2, we should privately test whether $S^0_{b_i}$ has a prefix of a_i in order to privately determine whether $a_i \geq b_i$. The test result can be got by *random-zero transformation* on a_i and $S^0_{b_i}$, and then a private scalar product:

1) For each bit a_{il} ($K \geq l \geq 1$) in a_i, Alice generates a *random-zero* pair (r_{l0}, r_{l1}): If $a_{il} = 1$, $r_{l0} \in_R \mathbb{Z}^*_q$ and $r_{l1} = 0$; if $a_{il} = 0$, $r_{l0} = 0$ and $r_{l1} \in_R \mathbb{Z}^*_q$. Then Alice gets a vector $\mathcal{R}_i = (r_{K0}, r_{K1}, ..., r_{10}, r_{11})$. For example, if $a_i = 0110$, $\mathcal{R}_i = (0, r_{41}, r_{30}, 0, r_{20}, 0, 0, r_{11})$.

2) For each bit $b[i,j,l]$ in $b[i,j]$ ($j = 1, ..., K$), Bob generates a random-zero pair (r'_{l0}, r'_{l1}): If $b[i,j,l] = 1$, $r'_{l0} = 0$, $r'_{l1} \in_R \mathbb{Z}^*_q$; if $b[i,j,l] = 0$, $r'_{l0} \in_R \mathbb{Z}^*_q$, $r'_{l1} = 0$.

   Let $|b[i,j]| = J_{ij}$, then Bob pads $K - J_{ij}$ pairs of $(0,0)$ at the end of these random-zero pairs, and gets a vector $\mathcal{R}'_{ij} = (r'_{K0}, r'_{K1}, ..., r'_{10}, r'_{11})$. For example, if $b[i,j] = 011$, $\mathcal{R}'_{ij} = (r'_{40}, 0, 0, r'_{31}, 0, r'_{21}, 0, 0)$.

3) Alice gives $E(\mathcal{R}_i) = (\ E(r_{K0}), E(r_{K1}), ..., E(r_{10}), E(r_{11})\ )$ to Bob. Bob computes $E(\mathcal{R}_i \cdot \mathcal{R}'_{ij}) = E(r_{K0})^{r'_{K0}} \odot E(r_{K1})^{r'_{K1}} \odot \cdots \odot E(r_{10})^{r'_{10}} \odot E(r_{11})^{r'_{11}} = E(\sum_{l=1}^{K}(r_{l0}r'_{l0} + r_{l1}r'_{l1}))$.

It's easy to see that if $b[i,j]$ is a prefix of a_i, $\mathcal{R}_i \cdot \mathcal{R}'_{ij} = 0$, and if $S^0_{b_i}$ has a prefix of a_i, $\prod_{j=1}^{K} \mathcal{R}_i \cdot \mathcal{R}'_{ij} = 0$. By Lemma 4 we also know that if $S^0_{b_i}$ has no prefix of a_i, $\prod_{j=1}^{K} \mathcal{R}_i \cdot \mathcal{R}'_{ij}$ can be zero with only negligible probability.

Lemma 4 *Let $\mathcal{R}_i = (r_{K0}, r_{K1}, ..., r_{10}, r_{11})$ be the random-zero transformation (over $\mathbb{Z}^*_q$) on a_i, and $\mathcal{R}'_{ij} = (r'_{K0}, r'_{K1}, ..., r'_{10}, r'_{11})$ be the random-zero transformation (over $\mathbb{Z}^*_q$) on $b[i,j]$ in $S^0_{b_i}$. If for $j = 1, ..., K$, $b[i,j]$ is not a prefix of a_i, $Pr[\prod_{j=1}^{K} \mathcal{R}_i \cdot \mathcal{R}'_{ij} = 0] \leq negl(\lambda)$.*

The proof of Lemma 4 is postponed to Appendix 2.10.2. By Lemma 4, to determine whether $A \succeq B$, Alice and Bob should know whether $\prod_{j=1}^{K} \mathcal{R}_i \cdot \mathcal{R}'_{ij} = 0$ for $i = 1, ..., n$. We disclose only $\mathcal{R} = \sum_{i=1}^{n}(\prod_{j=1}^{K}(\mathcal{R}_i \cdot \mathcal{R}'_{ij}))$ to both of them. If $a_i \geq b_i$ for $i = 1, ..., n$, all $\prod_{j=1}^{K} \mathcal{R}_i \cdot \mathcal{R}'_{ij} = 0$ and $\mathcal{R} = 0$. If there is any $i' \in \{1, ..., n\}$ such that $a_{i'} < b_{i'}$, by Lemma 5, we know that $\mathcal{R}$ is zero with only negligible probability.

Lemma 5 *For two vectors $A = (a_1, ..., a_n)$ and $B = (b_1, ..., b_n)$, let $\mathcal{R}_i = (r_{K0}, r_{K1}, ..., r_{10}, r_{11})$ be the random-zero transformation (over $\mathbb{Z}_q^*$) on a_i, and $\mathcal{R}'_{ij} = (r'_{K0}, r'_{K1}, ..., r'_{10}, r'_{11})$ be the random-zero transformation (over $\mathbb{Z}_q^*$) on $b[i, j]$ in $S^0_{b_i}$. Let $\mathcal{R} = \sum_{i=1}^{n}(\prod_{j=1}^{K}(\mathcal{R}_i \cdot \mathcal{R}'_{ij}))$. If there exists an $S^0_{b_{i'}}$ such that it has no prefix of $a_{i'}$, then: 1) $Pr[\mathcal{R} = 0] \leq negl(\lambda)$; 2) For $m = 1, ..., q-1$, $Pr[\mathcal{R} = m] \leq negl(\lambda)$.*

The proof of Lemma 5 is postponed to Appendix 2.10.2. By checking whether $\mathcal{R} = 0$, overwhelmingly we can determine whether $A \succeq B$, without disclosing the ordering of each pair (a_i, b_i) in the case of no dominance.

2.5.2 Privacy Preserving Product of Scalar Products

Given $E(\prod_{j=1}^{K}(\mathcal{R}_i \cdot \mathcal{R}'_{ij}))$ for $i = 1, ..., n$, it's easy to get the encrypted sum of them. However given each $E(\mathcal{R}_i \cdot \mathcal{R}'_{ij})$ for $j = 1, ..., K$, it's nontrivial to get the encrypted product $E(\prod_{j=1}^{K}(\mathcal{R}_i \cdot \mathcal{R}'_{ij}))$, because the encryption scheme we use is only additive homomorphic. We employ the block of *Privacy Preserving Product of Scalar Products* (PPPSP), to get $E(\prod_{j=1}^{K}(\mathcal{R}_i \cdot \mathcal{R}'_{ij}))$:

1) Alice sends $E(\mathcal{R}_i) = (E(r_{K0}), E(r_{K1}), ..., E(r_{11}))$ to Bob. Bob computes $E(P_1) = E(\mathcal{R}_i \cdot \mathcal{R}'_{i1}) = E(r_{K0})^{r'_{K0}} \odot E(r_{K1})^{r'_{K1}} \odot \cdots \odot E(r_{11})^{r'_{11}}$, in which $P_1 = \mathcal{R}_i \cdot \mathcal{R}'_{i1}$, and sends it to Alice.

2) For $j = 2, ..., K$,

   2.1) Alice computes $E(P_{j-1}\mathcal{R}_i) = (E(P_{j-1})^{r_{K0}} \odot E(0), E(P_{j-1})^{r_{K1}} \odot E(0), ..., E(P_{j-1})^{r_{11}} \odot E(0))$, and sends it to Bob.

   2.2) Bob computes $E(P_j) = E((P_{j-1}\mathcal{R}_i) \cdot \mathcal{R}'_{ij}) = E(P_{j-1}r_{K0})^{r'_{K0}} \odot E(P_{j-1}r_{K1})^{r'_{K1}} \odot \cdots \odot E(P_{j-1}r_{11})^{r'_{11}} = E(P_{j-1}(\mathcal{R}_i \cdot \mathcal{R}'_{ij}))$, in which $P_j = (P_{j-1}\mathcal{R}_i) \cdot \mathcal{R}'_{ij}$, and sends it to Alice.

3) Finally, they get $E((\mathcal{R}_i \cdot \mathcal{R}'_{i1}) \cdots (\mathcal{R}_i \cdot \mathcal{R}'_{iK}))$.

2.6 The Protocol for STVD in the Semi-honest Case

Our protocol for STVD in the semi-honest case is based on the ideas of random-zero transformation and private product of scalar products in Section 2.5. The protocol is applied in the multi-commodity private bidding:

Protocol 1 : STVD protocol for the semi-honest case

Input : Alice has a private bidding vector $A = (a_1, a_2, ..., a_n)$ ($a_i \neq 0$ for $i = 1, ..., n$) and Bob has a private sale price vector $B = (b_1, b_2, ..., b_n)$. Either Alice or Bob may be

semi-honest. Both of Alice and Bob hold the public key and their own shares of the secret key of the $(2,2)$-threshold ElGamal's cryptosystem.

Output : Both players output 1 if $A \succeq B$, and 0 otherwise, without gleaning any other information than the output.

Step 1 For $i = 1, ..., n$, Alice and Bob repeat the following:

1.1) *Random-zero transformation on a_i and b_i*: as described in Section 2.5.1, Alice gets a vector $\mathcal{R}_i = (r_{K0}, r_{K1}, ..., r_{10}, r_{11})$, Bob gets K vectors $\mathcal{R}'_{ij} = (r'_{K0}, r'_{K1}, ..., r'_{10}, r'_{11})$ for $j = 1, ..., K$.

1.2) *Privately Computing Product of Scalar Products*: as described in Section 2.5.2, Alice and Bob computes $E(\prod_{j=1}^{K}(\mathcal{R}_i \cdot \mathcal{R}'_{ij}))$.

Step 2 Alice and Bob compute $E(\mathcal{R}) = E(\prod_{j=1}^{K}(\mathcal{R}_1 \cdot \mathcal{R}'_{1j})) \odot \cdots \odot E(\prod_{j=1}^{K}(\mathcal{R}_n \cdot \mathcal{R}'_{nj})) = E(\ \sum_{i=1}^{n}(\ \prod_{j=1}^{K}(\mathcal{R}_i \cdot \mathcal{R}'_{ij})\ )\ )$.

Step 3 Alice and Bob cooperatively decrypt $E(\mathcal{R})$. If the decryption $g^{\mathcal{R}} = 1$, both of them output 1, otherwise, both of them output 0.

Theorem 1 *Protocol 1 achieves completeness and soundness for the problem of STVD when either of Alice and Bob is semi-honest.*

Proof: Let $\overline{M} = (M_1, M_2)$ be a pair of admissible strategies executed by Alice and Bob respectively in Protocol 1. Both parties follow the steps in Protocol 1. If $A \succeq B$, because $a_i \neq 0$ for $i = 1, ..., n$, by Lemma 2 $S^0_{b_i}$ has a prefix of a_i for $i = 1, ..., n$. Then it's easy to see that $E(\mathcal{R}) = E(0)$ and in Step 3, $g^{\mathcal{R}} = 1$. So an honest party M_k ($k \in \{1, 2\}$) outputs 1 with probability 1 and completeness is achieved by Definition 5.

If $A \not\succeq B$, by Lemma 2 there exists at least one $i \in \{1, ..., n\}$ such that $S^0_{b_i}$ has no prefix of a_i. By Lemma 5, $Pr(\mathcal{R} = 0) \leq negl(\lambda)$. Because g is the generator of G_q, then with the same negligible probability $g^{\mathcal{R}} = 1$. An honest party M_k outputs 1 with negligible probability and soundness is achieved by Definition 5. ■

Theorem 2 *Assuming the threshold ElGamal encryption is semantically secure, Protocol 1 is secure for the problem of STVD when either of Alice and Bob is semi-honest.*

Proof: Let $\overline{M} = (M_1, M_2)$ be the admissible strategies executed by Alice and Bob respectively in Protocol 1 denoted by Π, and $\overline{\mathcal{M}} = (\mathcal{M}_1, \mathcal{M}_2)$ be the admissible strategies executed by Alice and Bob in the ideal model.

If Alice is semi-honest, $REAL_{\Pi,\overline{M}}(A, B) = (M_1(A, \Pi_1(A, B)), \Pi_2(A, B))$, and $IDEAL_{f,\overline{\mathcal{M}}}(A, B) = (\mathcal{M}_1(A, f_1(A, B)), f_2(A, B))$. $M_1(A, \Pi_1(A, B)) = VIEW_{M_1}(A, \Pi_1(A, B), \mathcal{E}, g^{\mathcal{R}})$, i.e., Alice's view on $(A, \Pi_1(A, B), \mathcal{E}, g^{\mathcal{R}})$, in which $\mathcal{E}$ is the encryptions received by Alice. Due to the semantic security of the threshold ElGamal encryption (which is based on the DDH problem), $VIEW_{M_1}(A, \Pi_1(A, B), \mathcal{E}, g^{\mathcal{R}}) \equiv^c VIEW_{M_1}(A, \Pi_1(A, B), \mathcal{R}_E, g^{\mathcal{R}})$ in which $\mathcal{R}_E$ is a random sequence uniformly distributed over G_q.

As for $g^{\mathcal{R}}$, if $A \succeq B$, $g^{\mathcal{R}} = 1$. If $A \not\succeq B$, by Lemma 5, for $m = 0, 1, ..., q-1$, $Pr[\mathcal{R} = m] \leq negl(\lambda)$. Then for each $m' \in G_q$, $Pr[g^{\mathcal{R}} = m'] \leq negl(\lambda)$. Suppose $\mathcal{R}_R \in_R G_q$, then $Pr[g^{\mathcal{R}} = m'] - Pr[\mathcal{R}_R = m'] \leq negl(\lambda)$. Therefore $VIEW_{M_1}(A, \Pi_1(A, B), \mathcal{R}_E, g^{\mathcal{R}}) \equiv^c VIEW_{M_1}(A, \Pi_1(A, B), \mathcal{R}_E, \mathcal{R}_R)$.

In the ideal model, $\mathcal{M}_1(A, f_1(A, B)) \equiv^c VIEW_{\mathcal{M}_1}(A, f_1(A, B), \mathcal{R}'_E, \mathcal{R}'_R)$ in which $\mathcal{R}'_E$ is a random sequence uniformly chosen over G_q, and $\mathcal{R}'_R \in_R G_q$. By Theorem 1, $Pr[\Pi_2(A, B) = f_2(A, B)] \geq 1 - negl(\lambda)$, $Pr[\Pi_1(A, B) = f_1(A, B)] \geq 1 - negl(\lambda)$. Then $\mathcal{M}_1(A, f_1(A, B)) \equiv^c M_1(A, \Pi_1(A, B))$, and $IDEAL_{f,\overline{\mathcal{M}}}(A, B) \equiv^c REAL_{\Pi,\overline{M}}(A, B)$.

The case of a semi-honest Bob can be analyzed in the same way and the same result can be got, so Protocol 1 is secure with respect to semi-honest behaviors. ■

2.7 The Protocol for STVD in the Malicious Case

Protocol 1 may not be secure if one of the two parties is malicious, e.g., quit the protocol at any step or substitute the input arbitrarily. Protocol 1 should be fixed so that either each party is forced to behave in a semi-honest manner, or its malicious behaviors are detected by the other party. We fix Protocol 1 according to the malicious behaviors in multi-commodity private bidding where the bidder and seller act as Alice and Bob respectively.

2.7.1 Security against Malicious Bidder

In Step 1.1) of Protocol 1, a malicious bidder (Alice) may generate each pair (r_{l0}, r_{l1}) to be $(0, 0)$. Then whatever B is, finally $E(\mathcal{R}) = E(0)$, and the honest seller will always output $\Pi_2(A', B) = 1$. The malicious bidder may also generate each pair (r_{l0}, r_{l1}) in which $r_{l0} \neq 0$ and $r_{l1} \neq 0$, then with an honest seller, $Pr[\mathcal{R} = 0] \leq negl(\lambda)$, and $\Pi_2(A', B)$ is always 0. In this way the completeness and soundness of Protocol 1 will be easily breached. To detect these malicious behaviors, in Step 1.2), the bidder should send each pair $(E(r_{l0}), E(r_{l1}))$ to the seller and prove that (r_{l0}, r_{l1}) must be a random-zero pair with the proof 1) in Section 2.4.2. If the seller is not convinced, he will abort the protocol.

Protocol 1 has assumed that the bidding prices a_i for $i = 1, ..., n$ are nonzero. It's unnecessary to worry about that a malicious bidder would set some $a_i = 0$ in hope of catching the chance of $b_i = 0$ and winning the i-th commodity free of charge. In Protocol 1, if $a_i = 0$, whatever b_i is, by Lemma 1 $S^0_{b_i}$ can not have a prefix of a_i and thus the deal will not be made.

The malicious bidder may also arbitrarily replace her input A by $A' \in \{1, ..., 2^K - 1\}$, but in Theorem 3 we will show that, the possibility of gaining $\Pi_2(A', B) = 1$ (or 0) under the zero-knowledge proofs is the same with gaining $f_2(A', B) = 1$ (or 0) without apriori knowledge on B.

2.7.2 Security against Malicious Seller

In Step 1.1) of Protocol 1, a malicious seller (Bob) may also generate a vector $\mathcal{R}'_{ij}$ in which each entry is 0, or generate each pair of (r'_{l0}, r'_{l1}) in which $r'_{l0} \neq 0$ and $r'_{l1} \neq 0$. Then an honest seller will always output $\Pi_1(A, B') = 1$ or $\Pi_1(A, B') = 0$, and the completeness and soundness will also be breached. To detect these malicious behaviors, we notice that $b[i, j]$, the j-th element in $S^0_{b_i}$, must have a length $J_{ij} \geq j$ by Lemma 3. Then in the j-th iteration of Step 1.2), the seller should prove that the first j pairs (r'_{l0}, r'_{l1}) are random-zero pairs with the proof 3) in Section 2.4.2. In this way, the length of $b[i, j]$ is not leaked, and the seller can not also arbitrarily confirm the bidder that $\Pi_1(A, B') = 1$ or $\Pi_1(A, B') = 0$.

The malicious seller may also arbitrarily replace his input B by $B' \in \{0, ..., 2^K - 1\}$, but the possibility of gaining $\Pi_1(A, B') = 1$ (or 0) under the zero-knowledge proofs is the same with gaining $f_1(A, B') = 1$ (or 0) without apriori knowledge on A, which will be shown in Theorem 3.

2.7.3 The Protocol for the malicious case

Protocol 2 : STVD protocol for the malicious case

Input : Alice has a private bidding vector $A = (a_1, a_2, ..., a_n)$ and Bob has a private sale price vector $B = (b_1, b_2, ..., b_n)$. Either Alice or Bob may be malicious. Both of Alice and Bob hold the public key and their own shares of the secret key of the $(2, 2)$-threshold ElGamal's cryptosystem.

Output : Both players output 1 if $A \succeq B$, and 0 otherwise, without gleaning any other information than the output.

Step 1 For $i = 1, ..., n$, Alice and Bob repeat the following:

1.1) *Random-zero transformation on a_i and b_i*: as described in Section 2.5.1, Alice gets a vector $\mathcal{R}_i = (r_{K0}, r_{K1}, ..., r_{10}, r_{11})$, Bob gets K vectors $\mathcal{R}'_{ij} = (r'_{K0}, r'_{K1}, ..., r'_{10}, r'_{11})$ for $j = 1, ..., K$.

1.2) *Privately Computing Product of Scalar Products*

1.2.1) Alice sends $E(\mathcal{R}_i) = (C_{K0}, C_{K1}, ..., C_{11}) = (E(r_{K0}), E(r_{K1}), ..., E(r_{11}))$ to Bob, with the proof of $PK\{r_{l0}, r_{l1} | C_{l0} = E(r_{l0}) \wedge C_{l1} = E(r_{l1}) \wedge (r_{l0} r_{l1} = 0) \wedge (r_{l0} + r_{l1} \neq 0)\}$ for $l = 1, ..., K$, which is from the proof 1) in Section 2.4.2.

1.2.2) Bob computes $E(P_1) = E(\mathcal{R}_i \cdot \mathcal{R}'_{i1}) = E(r_{K0})^{r'_{K0}} \odot E(r_{K1})^{r'_{K1}} \odot \cdots \odot E(r_{11})^{r'_{11}}$, and sends it to Alice, with the proof:

$$PK\left\{ r'_{K0}, r'_{K1}, ..., r'_{11} \middle| \begin{matrix} E(P_1) = E(r_{K0})^{r'_{K0}} \odot E(r_{K1})^{r'_{K1}} \odot \cdots \odot E(r_{11})^{r'_{11}} \\ \wedge (\ r'_{K0} r'_{K1} = 0\ ) \wedge (\ r'_{K0} + r'_{K1} \neq 0\ ) \end{matrix} \right\},$$

which is from the proof 3) in Section 2.4.2.

1.2.3) For $j = 2, ..., K$, Alice and Bob repeat the following:

i) Alice computes $E(P_{j-1}\mathcal{R}_i) = (\ E(P_{j-1})^{r_{K0}} \odot E(0), E(P_{j-1})^{r_{K1}} \odot E(0), ..., E(P_{j-1})^{r_{11}} \odot E(0)\ )$, and sends it to Bob, with the proof of knowing the exponentials: $PK\{r_{l0}, r_{l1} | E(P_{j-1})^{r_{l0}} \odot E(0) \wedge C_{l0} = E(r_{l0}) \wedge E(P_{j-1})^{r_{l1}} \odot E(0) \wedge C_{l1} = E(r_{l1})\}$ for $l = 1, ..., K$, which is from the proof 2) in Section 2.4.2.

ii) Bob computes $E(P_j) = E(P_{j-1}\mathcal{R}_i \cdot \mathcal{R}'_{ij}) = E(P_{j-1} r_{K0})^{r'_{K0}} \odot E(P_{j-1} r_{K1})^{r'_{K1}} \odot \cdots \odot E(P_{j-1} r_{11})^{r'_{11}} = E(P_{j-1}(\mathcal{R}_i \cdot \mathcal{R}'_{ij}))$, and sends it to Alice, with the proof:

$$PK\left\{ r'_{K0}, r'_{K1}, ..., r'_{11} \middle| \begin{matrix} E(P_j) = E(P_{j-1} r_{K0})^{r'_{K0}} \odot E(P_{j-1} r_{K1})^{r'_{K1}} \odot \cdots \odot E(P_{j-1} r_{11})^{r'_{11}} \\ \bigwedge_{l=K-j+1}^{K} (\ r'_{l0} r'_{l1} = 0\ ) \bigwedge_{l=K-j+1}^{K} (\ r'_{l0} + r'_{l1} \neq 0\ ) \end{matrix} \right\},$$

which is from the proof 3) in Section 2.4.2.

Finally, Alice and Bob get $E(\prod_{j=1}^{K}(\mathcal{R}_i \cdot \mathcal{R}'_{ij}))$.

Step 2 Alice and Bob compute $E(\mathcal{R}) = E(\prod_{j=1}^{K}(\mathcal{R}_1 \cdot \mathcal{R}'_{1j})) \odot \cdots \odot E(\prod_{j=1}^{K}(\mathcal{R}_n \cdot \mathcal{R}'_{nj})) = E(\ \sum_{i=1}^{n}(\ \prod_{j=1}^{K}(\mathcal{R}_i \cdot \mathcal{R}'_{ij})\ )\ )$.

Step 3 Alice and Bob cooperatively decrypt $E(\mathcal{R})$. If the decryption $g^{\mathcal{R}} = 1$, both of them output 1, otherwise, both of them output 0.

Theorem 3 *Assuming the threshold ElGamal encryption is semantically secure and the zero-knowledge proofs in Protocol 2 are complete and sound, Protocol 2 achieves completeness and soundness for the problem of STVD when either of Alice and Bob is malicious.*

Proof: Let $\overline{M} = (M_1, M_2)$ be a pair of admissible strategies executed by Alice and Bob respectively in Protocol 2. We analyze the malicious behaviors of Alice and Bob respectively.

If M_1 is malicious, M_1 can generate $a'_i = M_1(a_i, \mathcal{E})$ in which $\mathcal{E}$ is the encryption sequence received before the i-th iteration of Step 1). Due to the semantic security of ElGamal encryption (which is based on the DDH problem), $a'_i = M_1(a_i)$, that is, M_1 can only generate an arbitrary $a'_i = a'_{iK}...a'_{i1}$ without apriori knowledge of M_2's input. Because of the completeness and soundness of the zero-knowledge proofs, in Step 1.2), M_1 must prove to M_2 that in every pair of (r_{l0}, r_{l1}), either $r_{l0} \neq 0$, $r_{l1} = 0$, or $r_{l0} = 0$, $r_{l1} \neq 0$. Though M_1 can intentionally set the value of r_{l0} or r_{l1} regardless of its randomness, Bob still generates a corresponding random-zero pair (r'_{l0}, r'_{l1}). Then from the analysis of Lemma 5, $Pr[\mathcal{R} = 0]$ will not be changed. That is, for $A' = (a'_1, ..., a'_n)$, if $A' \succeq B$, $Pr[M_2 = 1] = Pr[\mathcal{R} = 0] = 1$; if $A' \not\succeq B$, $Pr[M_2 = 1] = Pr[\mathcal{R} = 0] \leq negl(\lambda)$, so completeness and soundness hold, if M_1 is malicious.

Similar analysis can be done for the malicious M_2. In the j-th iteration of Step 1.2), a malicious M_2 can only: 1) arbitrarily generate $b[i,j]'$ from $\{0,1\}^j$ without apriori knowledge on A; 2) for $j \geq l \geq 1$, set r'_{l0} or r'_{l1} to some certain value, and the other one to zero. By Lemma 5, $Pr[\mathcal{R} = 0]$ will not be changed, that is, whether $\mathcal{R} = 0$ is still overwhelmingly determined by whether M_2 can generate a $b[i,j]'$ that is a prefix of a_i in each i-th iteration of Step 1). If for $i = 1, .., n$, there exists a $b[i,j]'$ that is a prefix of a_i, let $b'_i = b[i,j]'0...0$ with a suffix of $K - j$ zeros, then by Lemma 2 $a_i \geq b'_i$, $Pr[M_1 = 1] = Pr[\mathcal{R} = 0] = 1$, so completeness is achieved. If for some i, $b[i,j]'$ is not a prefix of a_i for $j = 1, ..., K$, M_2 does not choose b'_i such that $a_i \geq b'_i$, then $Pr[M_1 = 1] \leq negl(\lambda)$, and soundness is achieved. ■

Theorem 4 *Assuming the threshold ElGamal encryption is semantically secure, the zero-knowledge proofs in Protocol 2 are complete and sound, then Protocol 2 for the problem of Secure Two-party Vector Dominance is secure for Alice and Bob when either of them is a malicious party.*

Proof: Let $\overline{M} = (M_1, M_2)$ be a pair of admissible strategies executed by Alice and Bob respectively in Protocol 2, denoted by Π. Let $\overline{\mathcal{M}} = (\mathcal{M}_1, \mathcal{M}_2)$ be the admissible strategies executed by Alice and Bob in the ideal model. When either Alice or Bob is malicious, we construct $\mathcal{M}_k$ ($k \in \{1, 2\}$) using M_k as a subroutine (following the idea of proving security in the malicious model in [48]), then prove that the views of the two strategies are computationally indistinguishable.

If Alice is malicious, the major executions of $\mathcal{M}_2$ and $\mathcal{M}_1$ with a subroutine of M_1 are as follows:

1) For $i = 1, ..., n$, $\mathcal{M}_1$ and $\mathcal{M}_2$ repeat the following:

   1.1) M_1 intentionally generates a'_i and the corresponding $\mathcal{R}_i$. $\mathcal{M}_2$ generates $\mathcal{R}'_{ij}$ for $j = 1, ..., K$ based on its $S^0_{b_i}$.

1.2) $\mathcal{M}_1$ invokes M_1. The output of M_1 will be either aborting or $E(\mathcal{R}_i)$. When $\mathcal{M}_1$ gets $E(\mathcal{R}_i)$, she checks whether each pair of plaintexts (r_{l0}, r_{l1}) in $E(\mathcal{R}_i)$ is a random-zero pair. Specifically, $\mathcal{M}_1$ emulates the execution of the proof 1) in Section 2.4.2), checks whether an honest verifier would be convinced that $(E(r_{l0}), E(r_{l1}))$ is an encrypted random-zero pair, and decides whether it aborts.

1.3) If $\mathcal{M}_1$ has not aborted, she sends $\mathcal{R}_i$ to the *trusted third party* (TTP). The TTP computes $\mathcal{R}_i \cdot \mathcal{R}'_{i1}$ in which $\mathcal{R}'_{i1}$ is sent by $\mathcal{M}_2$, then sends $E(P_1) = E(\mathcal{R}_i \cdot \mathcal{R}'_{i1})$ to $\mathcal{M}_1$ and $\mathcal{M}_2$. $\mathcal{M}_1$ also sends $E(P_1)$ to M_1.

1.4) For $j = 2, ..., K$, $\mathcal{M}_1$ and $\mathcal{M}_2$ repeat the following:

1.4.1) $\mathcal{M}_1$ invokes M_1 to get $E(P_{j-1}\mathcal{R}_i) = (\ E(P_{j-1})^{r_{K0}} \odot E(0), E(P_{j-1})^{r_{K1}} \odot E(0), ..., E(P_{j-1})^{r_{11}} \odot E(0)\ )$. $\mathcal{M}_1$ checks whether an honest verifier would be convinced that each pair of exponentials (r_{l0}, r_{l1}) is the same pair encrypted in $E(\mathcal{R}_i)$ by emulating the proof 2) in Section 2.4.2).

1.4.2) If $\mathcal{M}_1$ has not aborted, she sends $\mathcal{R}_i$ to the TTP. The TTP computes $P_{j-1}\mathcal{R}_i \cdot \mathcal{R}'_{ij}$, then sends $E(P_j) = E(P_{j-1}\mathcal{R}_i \cdot \mathcal{R}'_{ij})$ to $\mathcal{M}_1$ and $\mathcal{M}_2$. $\mathcal{M}_1$ also sends $E(P_j)$ to M_1.

2) The TTP sends the final $\mathcal{R}$ to $\mathcal{M}_1$ and $\mathcal{M}_2$.

Then we need prove that the views of M_1 and $\mathcal{M}_1$ are computationally indistinguishable. Let $\mathcal{P}$ be the probability that M_1 convinces the honest M_2 that the pair of plaintexts (r_{l0}, r_{l1}) is a random-zero pair in the proof 1) of Section 2.4.2), then:

1) If $\mathcal{P} > negl(\lambda)$, by the soundness of the zero-knowledge proof, (r_{l0}, r_{l1}) must be a random-zero pair. In both the real and ideal model, with probability $\mathcal{P}$, the joint execution view is non-aborting and has the same input and output. In both models, with probability $1 - \mathcal{P}$, the joint execution is aborting and includes the same input and output.

2) If $\mathcal{P} \leq negl(\lambda)$, by the completeness of the zero-knowledge proof, $(E(r_{l0}), E(r_{l1}))$ is not an encrypted random-zero pair. In both models, with probability $1-\mathcal{P}$, the joint execution is aborting and includes the same input and output. With probability $\mathcal{P}$, M_1 can encrypt a pair of two zeros or two random numbers over $\mathbb{Z}_q^*$ without being detected, then influence $Pr[\mathcal{R} = 0]$. With probability $\mathcal{P}$, $\mathcal{M}_1$ sends a random-zero pair (r_{l0}, r_{l1}) to the TTP, and has no influence on $Pr[\mathcal{R} = 0]$. However, this inconsistency only negligibly occurs.

Similar analysis can be done when the proof 2) of Section 2.4.2) is used, and the joint execution views of $\overline{M}$ and $\overline{\mathcal{M}}$ are also computationally indistinguishable. Then $\{IDEAL_{f,\overline{\mathcal{M}}}(A', B)\} \equiv^c \{REAL_{\Pi,\overline{M}}(A', B)\}$, when Alice is malicious.

If Bob is malicious, $\mathcal{M}_2$ can be constructed using M_2 as a subroutine:

1) For $i = 1, ..., n$, $\mathcal{M}_1$ and $\mathcal{M}_2$ repeat the following:

1.1) $\mathcal{M}_1$ generates $\mathcal{R}_i$ based on its a_i. M_2 intentionally generates $b[i, j]'$ and the corresponding $\mathcal{R}'_{ij}$ for $j = 1, ..., K$. $\mathcal{M}_1$ sends $\mathcal{R}_i$ to the TTP, where $E(\mathcal{R}_i)$ is computed and sent to $\mathcal{M}_2$. $\mathcal{M}_2$ sends $E(\mathcal{R}_i)$ to M_2.

1.2) $\mathcal{M}_2$ invokes M_2 and gets $E(P_1') = E(r_{K0})^{r'_{K0}} \odot E(r_{K1})^{r'_{K1}} \odot \cdots \odot E(r_{11})^{r'_{11}}$. He checks whether an honest verifier would be convinced that the scalar product is correctly computed and the pair of exponentials (r'_{K0}, r'_{K1}) is a random-zero pair, by emulating the proof 3) of Section 2.4.2).

1.3) If $\mathcal{M}_2$ has not aborted, it sends $\mathcal{R}'_{i1}$ to the TTP. The TTP computes $P_1 = \mathcal{R}_i \cdot \mathcal{R}'_{i1}$, then sends $E(P_1)$ to $\mathcal{M}_1$ and $\mathcal{M}_2$. $\mathcal{M}_2$ also sends $E(P_1)$ to M_2.

1.4) For $j = 2, ..., K$, $\mathcal{M}_1$ and $\mathcal{M}_2$ repeat the following:

1.4.1) $\mathcal{M}_2$ invokes M_2 to get $E(P_j') = E(P_{j-1}r_{K0})^{r'_{K0}} \odot E(P_{j-1}r_{K1})^{r'_{K1}} \odot \cdots \odot E(P_{j-1}r_{11})^{r'_{11}}$. He checks whether an honest verifier would be convinced that the scalar product is correctly computed and each pair of exponentials (r'_{l0}, r'_{l1}) for $K \geq l \geq K - j + 1$ is a random-zero pair.

1.4.2) If $\mathcal{M}_2$ has not aborted, it sends $\mathcal{R}'_{ij}$ to the TTP. The TTP computes $P_{j-1}\mathcal{R}_i \cdot \mathcal{R}'_{ij}$, then sends $E(P_j) = E(P_{j-1}\mathcal{R}_i \cdot \mathcal{R}'_{ij})$ to $\mathcal{M}_1$ and $\mathcal{M}_2$. $\mathcal{M}_2$ also sends $E(P_j)$ to M_2.

2) The TTP sends the final $\mathcal{R}$ to $\mathcal{M}_1$ and $\mathcal{M}_2$.

Similar analysis can be done by the completeness and soundness of the proof 3) in Section 2.4.2), and we can get that the joint execution views of $\overline{M}$ and $\overline{\mathcal{M}}$ are computationally indistinguishable, i.e., $\{IDEAL_{f,\overline{\mathcal{M}}}(A, B')\} \equiv^c \{REAL_{\Pi,\overline{M}}(A, B')\}$, when Bob is malicious. The theorem holds. ■

2.8 Comparisons with other solutions

2.8.1 Complexities of Protocol 1 and Protocol 2

Complexity of Protocol 1 : We estimate the computation cost of our protocols from the number of *mod-exps* (modular exponentiations). Each ElGamal encryption has 2 values. The length of each value transferred between Alice and Bob is at most λ bits. Because our protocols and previous solutions have the same key generation step, we will not count the cost of key generation.

Firstly we calculate the mod-exps required in the block of PPPSP (in Section 2.5.2). In Step 1) of the block, because half of $\mathcal{R}_i$ are zeros, Alice computes $5K$ mod-exps, and sends $4K$ values to Bob. In the j-th round of Step 2.1), Alice computes $4K$ mod-exps (for a pair $(0, r)$ Alice can compute (g^s, h^s) and $E(P_{j-1})^r \odot (g^s, h^s)$), and sends $4K$ values. In the K rounds of the block, the maximum computation of Bob is $K^2 + 3K - 2$ mod-exps (the specific computation is shown in Appendix 2.10.3). Bob sends totally $2K$ values to Alice. Thus in PPPSP Alice and Bob maximumly compute $5K^2 + 4K - 2$ mod-exps, and the communication cost is $4K^2 + 2K$ values.

In Step 1 of Protocol 1 the block of PPPSP is repeated for n rounds. In Step 3 of Protocol 1 Alice and Bob concurrently compute one mod-exp and respectively send one value to decrypt $E(\mathcal{R})$. Therefore, in Protocol 1 Alice and Bob maximumly compute $n(5K^2 + 4K - 2) + 1$ mod-exps, and the communication cost is $n(4K^2 + 2K) + 2$ values.

K + 1 Parallel Execution of Protocol 1: Suppose Alice has K platforms for parallel execution and Bob has one platform, the total computation of Protocol 1 can be optimized by such kind of $K + 1$ parallel executions. To achieve high speed, Alice and

Bob can generate $(1,0)$ or $(0,1)$, instead of $(r,0)$ or $(0,r)$ in the block of random-zero transformation. Then in Step 2 of Protocol 1, Alice and Bob should compute $E(r_1 r_2 \mathcal{R})$ in which r_1 and r_2 are randomly generated by Alice and Bob respectively. In Step 3 if $g^{r_1 r_2 \mathcal{R}} = 1$, both of them output 1, otherwise output 0. Based on the correctness and security of Protocol 1, it is easy to validate the correctness and security of this speeded-up protocol. Then in a parallel execution of this protocol over $K+1$ platforms (as shown in Appendix 2.10.3), totally Alice and Bob need $n(4K+2)+1$ mod-exps .

In this parallel execution the communication cost is the same with the serial execution since the platforms of Alice communicate with those of Bob by a single line.

Complexity of Protocol 2 : Every proof of 1), 2) and 3) in Section 2.4.2) need $O(1)$ mod-exps, and exchanges $O(1)$ values (by [83, 37, 16, 58]). In the serial execution of Step 1.2) in Protocol 2, for each iteration of $j = 1, ..., K$, both Alice and Bob need $O(K)$ mod-exps. Thus, totally both Alice and Bob need $O(nK^2)$ mod-exps.

In $K+1$ parallel execution, each platform can compute the mod-exps to prove the corresponding (r_{l1}, r_{l0}) or (r'_{l1}, r'_{l0}) is a random-zero pair. Thus in parallel Alice and Bob in Protocol 2 totally need $O(nK)$ mod-exps.

In Step 1.2), for each iteration of $j = 1, ..., K$, Alice and Bob exchange $O(K)$ values. Therefore totally the communication cost between Alice and Bob is $O(nK^2\lambda)$ bits.

2.8.2 A Derived STVD Solution from [84]

Suppose each pair (a_i, b_i) has been put into a circuit for Yao's millionaire problem for $i = 1, ..., n$. The output of the circuit $E(t_i) = E(1)$ if $a_i > b_i$, and $E(t_i) = E(0)$ otherwise. To our knowledge the most efficient circuit of this kind is provided by [84], which need $12K$ mod-exps for each pair (a_i, b_i). Based on the conditional gate in [84], another STVD solution can be derived as following:

1) The two parties compute the output $E(t_i)$ by putting a_i and b_i into the circuit for $i = 1, ..., n$,

2) The two parties compute $E(\prod_{i=1}^{n} t_i)$ by supposing $T_1 = t_1$ and repeating the following for $i = 2, ..., n$:

   2.1) compute $E(t'_i) = E(2t_i - 1)$ in which $t'_i = 2t_i - 1 \in \{-1, 1\}$,

   2.2) compute $E(T_{i-1} t'_i)$ by conditional gate,

   2.3) compute $E(T_i) = E(T_{i-1} t_i) = E(\frac{T_{i-1} t'_i + T_{i-1}}{2})$, in which $T_i = T_{i-1} t_i$.

Step 1) need $12nK$ mod-exps. In step 2.2) one conditional gate need 23 mod-exps (according to [84]). In step 2.3) the quadratic residues need 2 mod-exps on the ElGamal encryption $E(T_{i-1} t'_i + T_{i-1})$. Thus, in the serial execution, the derived STVD solution totally need $12nK + 25(n-1)$ mod-exps.

In $K+1$ parallel execution, because Bob has only one platform, in step 1), the two parties have to compute $E(t_i)$ for $i = 1, ..., n$ one by one. In step 2), the two parties also have to compute $E(T_i)$ for $i = 2, ..., n$ one by one. Therefore, $K+1$ parallel execution will not speed up the derived solution in the computation cost.

$8K$ values need be transferred between Alice and Bob in the circuit of [84]. 12 values need be transferred in their conditional gate. Therefore the total communication cost of the derived solution is $8nK + 12(n-1)$ values.

For the malicious case, some zero-knowledge proofs are added into the derived solution, and each of them need $O(1)$ mod-exps and transfer $O(1)$ values. Thus the total costs of the derived solution are $O(nK)$ mod-exps, and $O(nK\lambda)$ communication bits for the serial and $K+1$ executions.

2.8.3 Comparisons

We summarize the above complexities in Table 2.1. In the table we also compute the mod-exps and communication bits of the STVD protocol in [4] which is based on semi-honest model (The details of the computation can be found in the Appendix 2.10.3).

In multi-commodity bidding, practically K can be set as 5 to 10. As pointed out in [71], for long-term security a 1024-bit modulus should be considered, so we set $\lambda = 1024$. We have also tested the modular exponentiations on a computer with a CPU of 2.0GHz (Pentium 4), and the running time of one 1024-bit mod-exp is about 5 milliseconds.

In serial executions Protocol 1 has no significant increase on the mod-exps compared with the two previous solutions. For example, let $K = 10$, $n = 10$, then the mod-exps costs of Protocol 1, the derived solution from [84] , and the solution from [4] are about 26, 7 and 26 seconds respectively.

In $K+1$ parallel executions, Protocol 1 achieves superiority over the efficiency of the solutions from [84] and [4]. For example, let $K = 10$, $n = 20$, then the mod-exps costs of Protocol 1, the solution from [84], and the solution from [4] are about 4.2, 14.2 and 49.7 seconds respectively. In $K+1$ parallel executions, Protocol 2 has the same level of computation cost with the derived solution for the malicious case.

Suppose the networks of Alice and Bob are linked by a T3 line (32Mbps), the communication costs of the 5 protocols will not be significant overloads. For example, let $K = 10$, $n = 20$, the communication bits of Protocol 1, the solution from [84], and the solution from [4] can be completed in 0.26, 0.06 and 0.21 seconds.

Table 2.1: Comparisons for STVD Solutions

		Computation Cost (mod-exps)		*Communication bits*
		Serial	$K+1$ *Parallel*	
Semi-honest	*Protocol 1*	$n(5K^2+4K-2)+1$	$4nK+2n+1$	$[n(4K^2+2K)+2]\lambda$
	[84]	$12nK+25(n-1)$	$12nK+25(n-1)$	$[8nK+12(n-1)]\lambda$
	[4]	$48nK+40n$	$48nK+16n+8\lceil\frac{n}{K}\rceil$	$(32nK+16n)\lambda$
Malicious	*Protocol 2*	$O(nK^2)$	$O(nK)$	$O(nK^2\lambda)$
	[84]	$O(nK)$	$O(nK)$	$O(nK\lambda)$

2.9 Conclusions

In this section we define the problem of secure two-party vector dominance, and provide STVD protocols based on an additive homomorphic encryption. A protocol is firstly constructed in the semi-honest model, and then extended for the malicious model. Correctness and security of the protocols are analyzed for both models. The protocols can be used in applications such as multi-commodity private bidding to securely negotiate a deal on a series of commodities.

We also provide two ways to execute our protocols: serial and $K+1$ parallel executions. When there are enough moderately fast platforms, the parallel execution can be used to achieve less costs than the serial execution. For the semi-honest case our protocol can achieve superiority over the efficiency of the solutions from [84] and [4] by the parallel execution. For the malicious case our protocol has the same level of computation cost with the solution from [84]. Our protocols need more communication bits, but they can be completed by no more than one second in practical applications, therefore the bits will not decrease the superiority of our protocols.

2.10 Appendix

2.10.1 Major Notations

n	the dimension of vector A and B, $A = \{a_i \| i = 1, ..., n\}$, $B = \{b_i \| i = 1, ..., n\}$
K	the length of each a_i and b_i
λ	the length of modulus p in ElGamal encryption
a_{il}	the l-th bit in a_i from the right, $K \geq l \geq 1$
$S^0_{b_i}$	the 0-encoding list of b_i
$b[i, j]$	the j-th element in $S^0_{b_i}$
$\|b[i, j]\|$	the length of $b[i, j]$
L_{ij}	the length of $b[i, j]$
J_i	the number of "0" bits in b_i

2.10.2 Proofs of Lemmas for Building Blocks

Lemma 4 *Let $\mathcal{R}_i = (r_{K0}, r_{K1}, ..., r_{10}, r_{11})$ be the random-zero transformation (over $\mathbb{Z}^*_q$) on a_i, and $\mathcal{R}'_{ij} = (r'_{K0}, r'_{K1}, ..., r'_{10}, r'_{11})$ be the random-zero transformation (over $\mathbb{Z}^*_q$) on $b[i, j]$ in $S^0_{b_i}$. If for $j = 1, ..., K$, $b[i, j]$ is not a prefix of a_i, $Pr[\prod_{j=1}^{K} \mathcal{R}_i \cdot \mathcal{R}'_{ij} = 0] \leq negl(\lambda)$.*

Proof: $\mathcal{R}_i \cdot \mathcal{R}'_{ij} = \sum_{l=K-J_{ij}+1}^{K} (r_{l0} r'_{l0} + r_{l1} r'_{l1})$. Let $r_l = r_{l0} r'_{l0} + r_{l1} r'_{l1}$. If $b[i, j]$ is not a prefix of a_i, there exists $b[i, j, l] \neq a_{il}$ for some $l \in \{K - J_{ij} + 1, ..., K\}$, and r_l is either $r_{l0} r'_{l0}$ or $r_{l1} r'_{l1}$ but nonzero. W.l.o.g. $r_l = r_{l0} r'_{l0}$, because r_{l0}, $r'_{l0} \in_R \mathbb{Z}^*_q$ and $\mathbb{Z}^*_q$ is a multiplicative subgroup of $\mathbb{Z}_q$, r_l is also uniformly distributed over $\mathbb{Z}^*_q$ ($r_l \neq 0$). That is, $Pr[r_l = m] = \frac{1}{q-1}$ for $m \in \{1, ..., q-1\}$.

Suppose there are S_j ($1 \leq S_j \leq J_{ij}$) bits in $b[i, j]$ such that $b[i, j, l] \neq a_{il}$ for $l \in \{K - J_{ij} + 1, ..., K\}$, then $\mathcal{R}_i \cdot \mathcal{R}'_{ij} = \sum_{l'=1}^{S_j} r_{l'}$. Let $Pr[\sum_{l'=1}^{S_j} r_{l'} = 0] = P_{S_j}$, then

$$
\begin{aligned}
P_{S_j} &= Pr[\sum_{l'=1}^{S_j - 1} r_{l'} = 0] Pr[r_{S_j} = 0] + \sum_{m=1}^{q-1} Pr[\sum_{l'=1}^{S_j - 1} r_{l'} = m] Pr[r_{S_j} = q - m] \\
&= 0 + (q-1) \cdot \frac{1 - P_{S_j - 1}}{q-1} \cdot \frac{1}{q-1}
\end{aligned}
$$

i.e,. $\frac{1}{q} - P_{S_j} = -\frac{1}{q-1}(\frac{1}{q} - P_{S_j - 1})$. Given $P_1 = 0$, $P_{S_j} = \frac{1}{q}(1 - \frac{1}{(1-q)^{S_j - 1}})$. It's easy to see

that $Max(P_{S_j}) = P_2 = \frac{1}{q-1}$. Then

$$Pr[\prod_{j=1}^{K} \mathcal{R}_i \cdot \mathcal{R}'_{ij} = 0] = Pr[\mathcal{R}_i \cdot \mathcal{R}'_{i1} = 0] + Pr[\mathcal{R}_i \cdot \mathcal{R}'_{i1} \neq 0]Pr[\mathcal{R}_i \cdot \mathcal{R}'_{i2} = 0] + ...$$
$$+ Pr[\mathcal{R}_i \cdot \mathcal{R}'_{i1} \neq 0] \cdots Pr[\mathcal{R}_i \cdot \mathcal{R}'_{i(K-1)} \neq 0]Pr[\mathcal{R}_i \cdot \mathcal{R}'_{iK} = 0]$$
$$= P_{S_1} + (1 - P_{S_1})P_{S_2} + ... + \prod_{j=1}^{K-1}(1 - P_{S_j})P_{S_K}$$

Because $(1 - P_{S_1})P_{S_2} < P_{S_2}$, ..., $\prod_{j=1}^{K-1}(1 - P_{S_j})P_{S_K} < P_{S_K}$, $P_{S_j} \leq \frac{1}{q-1}$ for $j = 1, ..., K$, then

$$Pr[\prod_{j=1}^{K} \mathcal{R}_i \cdot \mathcal{R}'_{ij} = 0] < K \cdot \frac{1}{q-1}$$
$$= \frac{K}{2^{\Omega(\lambda)}}$$
$$\leq negl(\lambda)$$

Then the lemma holds. ■

Lemma 5 *For two vectors $A = (a_1, ..., a_n)$ and $B = (b_1, ..., b_n)$, let $\mathcal{R}_i = (r_{K0}, r_{K1}, ..., r_{10}, r_{11})$ be the random-zero transformation (over $\mathbb{Z}_q^*$) on a_i, and $\mathcal{R}'_{ij} = (r'_{K0}, r'_{K1}, ..., r'_{10}, r'_{11})$ be the random-zero transformation (over $\mathbb{Z}_q^*$) on $b[i, j]$ in $S^0_{b_i}$. Let $\mathcal{R} = \sum_{i=1}^{n}(\prod_{j=1}^{K}(\mathcal{R}_i \cdot \mathcal{R}'_{ij}))$. If there exists an $S^0_{b_{i'}}$ such that it has no prefix of $a_{i'}$, then: 1) $Pr[\mathcal{R} = 0] \leq negl(\lambda)$; 2) For $m = 1, ..., q-1$, $Pr[\mathcal{R} = m] \leq negl(\lambda)$.*

Proof: Suppose there exists T $(1 \leq T \leq n)$ $b_{i'}$ such that $S^0_{b'_i}$ has no prefix of $a_{i'}$, let $\mathcal{R} = R_T = \sum_{i'=1}^{T}(\prod_{j=1}^{K}(\mathcal{R}_{i'} \cdot \mathcal{R}'_{i'j}))$, and $Pr[\mathcal{R} = 0] = Pr[R_T = 0] = P_T$. Then

$$P_T = Pr[R_{T-1} = 0]Pr[\prod_{j=1}^{K}(\mathcal{R}_{i'} \cdot \mathcal{R}'_{i'j}) = 0] + \sum_{m=1}^{q-1} Pr[R_{T-1} = m]Pr[\prod_{j=1}^{K}(\mathcal{R}_{i'} \cdot \mathcal{R}'_{i'j}) = q - m]$$
$$= P_{T-1}P_1 + (q-1) \cdot \frac{(1 - P_{T-1})}{q-1} \cdot \frac{1 - P_1}{q-1}$$
$$= \frac{qP_1 - 1}{q-1}P_{T-1} + \frac{1 - P_1}{q-1}$$

Then $P_T - \frac{1}{q} = \frac{qP_1-1}{q-1}(P_{T-1} - \frac{1}{q})$, and $P_T = (P_1 - \frac{1}{q})(\frac{qP_1-1}{q-1})^{T-1} + \frac{1}{q}$.
By Lemma 4, $P_1 = Pr[R_1 = 0] < \frac{K}{q-1}$. It's easy to see that $\frac{qP_1-1}{q-1} < 1$, so for $1 \leq T \leq n$,

$$P_T \leq (P_1 - \frac{1}{q}) + \frac{1}{q} = P_1 \leq negl(\lambda)$$

For $m = 1, ..., q-1$,

$$Pr[\mathcal{R} = m] = \frac{1 - Pr[\mathcal{R} = 0]}{q-1} < \frac{1}{q-1} = 2^{-\Omega(\lambda)} \leq negl(\lambda)$$

The lemma holds. ■

2.10.3 Some calculations on complexities

The maximum computation of Bob in the block of PPPSP: In the block of PPPSP, the mod-exps that Bob need can be reduced by the lengths of all $b[i,j]$ in $S^0_{b_i}$, which depend on the number of zero bits in b_i. If the length $|b[i,j]| < K$, some $(0,0)$ pairs are padded in $\mathcal{R}'_{ij}$, then Bob need not compute mod-exps on these $(0,0)$ pairs. When $j = J_i + 1$ (J_i is the number of zero bits in b_i), $b[i,j] = b_i$ and $|b[i,j]| = K$, then Bob computes $2K$ mod-exps to get $E(P_{J_i+1})$. For $j = J_i + 2, ..., K$, $b[i,j] = b_i$, then Bob can compute only $E(P_{J_i+1}) \odot (g^s, h^s)$ and return it, without compromising the correctness and security of Protocol 1.

Thus, it's easy to validate that Bob has minimum mod-exps when there is no zero bit in b_i ($b_i = (1...1)_2$), and maximum mod-exps when only the first bit of b_i is zero ($b_i = (10...0)_2$). For the latter case, $S^0_{b_i} = \{(11)_2, (101)_2, ..., (10...1)_2, (10...0)_2\}$. Then for $b[i,1] = (11)_2$, Bob generates $(0, r, 0, r)$ and computes 4 mod-exps to get $E(P_1)$. For $b[i, K-1] = (10...1)_2$, Bob computes $2K$ mod-exps to get $E(P_{K-1})$. For $b[i,K] = (10...0)_2$, $2K$ mod-exps are also computed to get $E(P_K)$. Then the total mod-exps of Bob are $4 + 6 + ... + 2K + 2K = K^2 + 3K - 2$.

$K + 1$ **Parallel Execution of Protocol 1**: In Step 1) of PPPSP, Alice computes $E(1), E(0)$ on the l-th platform for $l = 1, ..., K$, each of which need 4 mod-exps. Bob computes $E(P_1) = E(\mathcal{R}_i \cdot \mathcal{R}'_{i1}) \odot E(0)$ and need only 2 mod-exps. In the j-th round of Step 2.1) of PPPSP, Alice computes 4 mod-exps on each platform to get $E(0) = (g^{s_1}, h^{s_1})$ and $E(P_{j-1}) \odot (g^{s_2}, h^{s_2})$. (Actually in the $(j-1)$-th round, when Bob is computing $E(P_{j-1})$, on each platform Alice can compute the 2 mod-exps for $E(0)$ which will be used in the j-th round in advance. Then when the j-th round comes, on each platform Alice need only compute the multiplication $E(P_{j-1}) \odot (g^{s_2}, h^{s_2}))$. In the j-th round Bob computes $E(P_j) = E((P_{j-1}\mathcal{R}_i) \cdot \mathcal{R}'_{ij}) \odot E(0)$ and need only 2 mod-exps. Thus in parallel, totally Alice and Bob in PPPSP need $4K + 2$ mod-exps. Including the cost of Step 3 in Protocol 1, totally Alice and Bob need $n(4K + 2) + 1$ mod-exps.

The cost of STVD Solution from [4] There are 4 blocks in this STVD solution: 1) input disguise, 2) private permutation, 3) Yao's millionaire computation, and 4) private equality-testing. The first block need not mod-exps and communication bits. We suppose the second block uses Paillier's encryption scheme, the third block uses the efficient circuit from Schoenmakers's paper, and the fourth block uses Pohlig-Hellman encryption scheme.

1) *Serial Execution*: In the first block Alice and Bob extend their vectors to be $4n$ elements respectively. It's easy to compute that the cost of the second block is $24n$ mod-exps, and the communication cost is $8n$ values. The third block need $4n$ circuits from [84], thus need $4n * 12K = 48nK$ mod-exps and transfer $32nK$ values. The fourth block need $16n$ mod-exps and transfer $8n$ values. Thus totally Alice and Bob need $48nK + 40n$ mod-exps, and transfer $32nK + 16n$ mod-exps.

2) $K + 1$ *Parallel Execution*: In this kind of execution, Alice's computation can be executed in parallel, but Bob's can not. In the second block the two sides need $8n + 4\lceil \frac{n}{K} \rceil$ mod-exps. In the third block the two sides need $48nK$ mod-exps. In the fourth block the two sides need $8n + 8\lceil \frac{n}{K} \rceil$ mod-exps. Thus totally the two sides need $48nK + 16n + 8\lceil \frac{n}{K} \rceil$ mod-exps.

The communication bits of the two parallel executions are the same with the serial execution since we assume the networks of Alice and Bob are linked by a single line.

2.10.4 Some Basic Zero-knowledge Proofs

Proving Knowing the Discret Log : $PK\{\alpha|C = g^{\alpha} \ mod \ p\}$. For the proof, the prover and verifier run the following steps (from [83]):

1 The prover selects $r \in_R \{1, ..., p-1\}$, computes $x = g^r \ mod \ p$, and sends x to the verifier.

2 The verifier selects a random number $e \in_R \{0, ..., 2^t - 1\}$, and sends e to the prover.

3 The prover computes $y = r + \alpha e \ mod \ \varphi(p)$, and sends y to the verifier.

4 The verifier checks whether $g^y = xC^e$.

Proving Knowing Two Discret Logs : $PK\{x, r|C = b_0^x b_1^r \ mod \ p\}$. The prover should prove that he knows (x, r) such that $C = b_0^x b_1^r \ mod \ p$. The proof is as follows (from [37]) :

1 The prover chooses w_1^0, $w_1^1 \in_R [0, 2^{2m}p)$ and sets w_2^0, w_2^1 by $w_2^0 = w_1^0 - 2^{2m}p$ and $w_2^1 = w_1^1 - 2^{2m}p$. The prover picks four elements, $w_{i,j}^2 \in_R [0, 2^{2m}p)$, then computes $t_{i,j} = b_0^{w_i^0} b_1^{w_j^1} b_2^{w_{i,j}^2}$, where $1 \leq i, j \leq 2$.

2 The prover sends to the verifier, four unordered commitments, $t_{i,j}$'s.

3 The verifier picks a challenge $c \in_R [0, 2^{2m})$ and sends it to the prover.

4 The prover sets $X = cx + w_i^0$ and $R = cr + w_j^1$ such that $X, R \in [0, 2^{2m}p)$, and sends to the verifier, the pair, $(X, R, w_{i,j}^2)$.

5 The verifier checks there exists a $t_{i,j}$ such that $b_0^X b_1^R b_2^{w_{i,j}^2} = t_{i,j} C^c \ mod \ p$.

Proving Correct Multiplication : $PK\{a, r, t, b, u|A = h^a g^r, B = h^b g^u, C = B^a g^t\}$. This proof is from [16]. The protocol proceeds by executing the following two 3-step protocols in parallel, using the sam challenge e in both instances. The first is intended to verify that A, C have the correct form, while the second verifies that the prover can open B:

1. First protocol:

   1.1) The prover chooses $x, s_1, s_2 \in_R \mathbb{Z}_q$ and sends $m_1 = h^x g^{s_1}$, $m_2 = B^x g^{s_2}$ to the verifier.

   1.2) The verifier chooses a random number e, so that $0 \leq e < q$ and sends it to the prover.

   1.3) The prover sets $z = x + ae \ mod \ q$ and $w_1 = s_1 + re \ mod \ q$ and $w_2 = s_2 + te \ mod \ q$. He sends z, w_1, w_2 to the verifier, who verifies that $h^z g^{w_1} = m_1 A^e$ and $B^z g^{w_2} = m_2 C^e$.

2. Second protocol:

   2.1) The prover chooses $d, s \in_R \mathbb{Z}_q$, and sends $m = h^d g^s$ to the verifier.

2.2) The verifier chooses a random number e, so that $0 \leq e < q$ and sends it to the prover.

2.3) The prover sets $z = d + eb \; mod \; q$. He then computes $w = s + ue$, and sends z, w to the verifier, who verifies that $h^z g^w = mB^e$.

Chapter 3

Privacy Preserving Set Intersection among Multiple Parties

3.1 Introduction

For datasets distributed on different sources, intersection among these sets is always required to gain useful information. For example, supermarkets need find out the same card numbers which have consuming records in all of their databases, and then provide better service for the card owners. For such kind of applications, privacy may be a critical concern of the data owners, so they are reluctant to directly publish their datasets. Specifically, one supermarket doesn't want other supermarkets to know the card numbers in its database except those in the intersection. Therefore, there should be some privacy preserving techniques for them to determine the results of set intersection, without the datasets being directly published.

In this chapter, we address the problem of *Privacy Preserving Set Intersection* (PPSI), in which there are N ($N \geq 2$) parties, each party P_i ($i = 1, ..., N$) has a set (or multiset) T_i and $|T_i| = S$, all parties want to learn the intersection $TI = T_1 \cap ... \cap T_N$, without gleaning any information other than those computed from a coalition of parties inputs and outputs. Basically, we solve PPSI by efficiently constructing and evaluating polynomials whose roots are elements of the set intersection.

The contributions of this chapter include:

1) We give formal definitions of PPSI in both the semi-honest and malicious models.

2) We propose an efficient PPSI protocol for the semi-honest model, which has lower computation and communication costs than the PPSI protocols in [62] and [36].

3) We improve the PPSI protocol in the semi-honest model to be secure in the malicious model, and prove its security in the malicious model by the definition. A PPSI protocol for the malicious model has also been proposed in [63], but our protocol achieves lower computation and communication costs in comparisons.

The remainder of the chapter is organized as follows: Section 3.2 discusses some related work. The problem of PPSI is defined in Section 3.3. Section 3.4 lists the basic tools for our protocol. Section 3.5 and 3.6 propose the PPSI protocol for the semi-honest and malicious model respectively. In Section 3.7 we analyze the security of our PPSI

protocol. In Section 3.8 we compare our protocols with the related work considering the computation and communication costs. Section 3.9 concludes the whole chapter. In the appendix of Section 3.10 we give the detailed proofs of two theorems of this chapter.

3.2 Related Work

PPSI can be traced back to the specific problem of private equality test (PET) in two-party case, where each party has a single element and wants to test whether they are equal without publishing the elements. The problem of PET was considered in [9], [29], [67] and [74]. PET solutions can't be simply used for the multi-party cases of PPSI, otherwise too much sensitive information will be leaked, e.g., any two parties will know the intersection of their private sets.

A solution for the multi-party case of PPSI was firstly proposed in [36]. The solution is based on evaluating polynomials representing elements in the sets. In [62], another solution for PPSI was proposed, in which each polynomial representing each set is multiplied by a random polynomial which has the same degree with the former polynomial. In this chapter, to get a solution with lower cost than [36] and [62], we multiply each polynomial representing each set by a random polynomial which has a low enough degree without compromising the security of the solution. We also multiply the randomized polynomials by a nonsingular matrix to improve the correctness of our solution. We will compare our solution for PPSI with [36] and [62] in details in Section 3.8.

The PPSI protocol in the semi-honest model in [62] was fixed to be secure in the malicious model in [63]. We will also extend our PPSI protocol in the semi-honest model to the malicious model, and compare it with the work in [63].

3.3 Problem Definition

Our PPSI problem aims to securely compute the set intersection among N ($N \geq 2$) parties. Generally speaking there are two types of probabilistic polynomial-time (PPT) bounded adversaries in SMC: semi-honest and malicious, as pointed out in Chapter 1.

Suppose all sets held by the parties are subsets of a common set $\mathbb{T}$, firstly we should prevent the dictionary attack, in which an adversary may defraud the honest party of inputs using $\mathbb{T}$. Therefore, we assume that each party holds a set (or multiset) of the same size S and $S \ll |\mathbb{T}|$, such that given two arbitrarily selected subsets T_i and $T_{i'}$, the probability that an input $a \in T_i$ equals any input $a' \in T_{i'}$ is negligible (i.e., $\frac{S}{|\mathbb{T}|} \rightarrow 0$).

Let N ($N \geq 2$) be the number of parties, each party P_i ($i = 1, ..., N$) has a set (or multiset): $T_i = \{T(i,j)|j = 1, ..., S\}$. Let $TI = T_1 \cap ... \cap T_N$. Let $\mathfrak{f}$ be an N-ary function:

$$\mathfrak{f}(T_1, ..., T_N) = \mathfrak{f}(\overline{T}) = \{\mathfrak{f}_{ij}(\overline{T})|\mathfrak{f}_{ij} \in \{0,1\}, i = 1, ..., N, j = 1, ..., S\}, in\ which$$
$$\mathfrak{f}_{ij}(\overline{T}) = 1\ if\ T(i,j) \in TI,$$
$$\mathfrak{f}_{ij}(\overline{T}) = 0\ if\ T(i,j) \notin TI.$$

Suppose $I = \{i_1, ..., i_c\} \subset \{1, ..., N\}$ be the index set of c ($1 \leq c \leq N-1$) colluded parties controlled by an adversary. Let $f_i(\overline{T}) = \{f_{ij}(\overline{T})|j = 1, ..., S\}$ and $\mathfrak{f}_I(\overline{T}) = \{\mathfrak{f}_{i_1}(\overline{T}), ..., \mathfrak{f}_{i_c}(\overline{T})\}$. Below we define the problem of privacy preserving set intersection in the semi-honest model and malicious model respectively.

Definition 6 (**PPSI in the semi − honest model**) *Let* Π *be an* N*-party protocol for computing* $\mathfrak{f}$*. Let* $VIEW_i^{\Pi}(\overline{T})$ *denote the view of the* i*-th party on its input, output, randomness and public transcript during an execution of* Π*. Let* $VIEW_I^{\Pi}(\overline{x}) = (I, VIEW_{i_1}^{\Pi}(\overline{T}), ..., VIEW_{i_c}^{\Pi}(\overline{T}))$.

Π *is said to privately solve the problem of Privacy Preserving Set Intersection with respect to the semi-honest behavior, if there exists a PPT algorithm* S*, such that*

$$\{S(I, (T_{i_1}, ..., T_{i_c}), \mathfrak{f}_I(\overline{T}))\} \equiv^c \{VIEW_I^{\Pi}(\overline{T})\} \tag{3.1}$$

Definition 7 (**PPSI in the malicious model**) *Let* Π *be an* N*-party protocol for computing* $\mathfrak{f}$*. Let a pair* (I, A)*, where* A *is a PPT algorithm, represent an adversary in the real model. The joint execution of* Π *under* (I, A) *in the real model, denoted* $REAL_{\Pi,I,A}(\overline{T})$*, is defined as the output sequence resulting from the interaction among the* N *parties in the execution of* Π*.*

Let a pair (I, B)*, where* B *is a PPT algorithm, represent an adversary in the ideal model, where there is an available trusted third party. The joint execution of* $\mathfrak{f}$ *under* (I, B) *in the ideal model, denoted* $IDEAL_{\mathfrak{f},I,B}(\overline{T})$*, is defined as the output pair of* B *and the honest parties in the ideal execution.*

Π *is said to securely solve the problem of privacy preserving set intersection in the malicious model, if for every PPT algorithm* A *(representing a real-model adversary strategy), there exists a PPT algorithm* B *(representing an ideal-model adversary strategy), such that*

$$\{IDEAL_{\mathfrak{f},I,B}(\overline{T})\} \equiv^c \{REAL_{\Pi,I,A}(\overline{T})\}. \tag{3.2}$$

It should be pointed out that both definitions of security have implied the correctness of Π. The views of parties in the real execution of Π should be computationally indistinguishable from their views in the ideal model, so it is necessary that the output of Π on $\overline{T}$ should also be computationally indistinguishable from the output of $\mathfrak{f}$ on $\overline{T}$.

3.4 Basic Tools

3.4.1 Homomorphic Encryption

Our protocol need a threshold and additive Homomorphic Encryption (HE) scheme. Thus, we use Paillier's cryptosystem ([75]) for its following properties:

1) it is an additive homomorphic encryption scheme. Given two encryptions $E(m_1)$ and $E(m_2)$, $E(m_1 + m_2) = E(m_1) \cdot E(m_2)$;

2) given an encryption $E(m)$ and a scalar a, $E(a \cdot m) = E(m)^a$;

3) (N, N)-threshold decryption can be supported (by [33],[34]).

In this chapter, $\mathcal{N}$ is the RSA-modulus which is the multiplication of two large prime numbers, and $\mathbb{Z}_{\mathcal{N}}$ is the plaintext space of Paillier's cryptosystem. A (l, t)-*threshold decryption* scheme is a protocol that allows any subset of $t+1$ out of l entities, or servers, to decrypt a ciphertext, but disallows the decryption if less than t servers participate in the protocol. Let $\Delta = l!$, below we give a description of the (l, t)-threshold version of Paillier cryptosystem.

- **Key Generation** : Choose an integer $\mathcal{N}$, product of two safe primes p and q, such that $p = 2p' + 1$ and $q = 2q' + 1$ and $gcd(n, \varphi(n)) = 1$. One can note that the safe prime requirement can be avoided ([35]) using Shoup protocol ([86]) without using safe primes. This allows to fully share Paillier cryptosystem from the key generation protocol to the decryption process as it appears difficult to generate RSA moduli with safe prime modulus using [8]. However, for the clarity of the description we use RSA moduli with safe primes.

  Set $m = p'q'$. Let β be an element randomly chosen in $\mathbb{Z}^*_{\mathcal{N}}$. The public key PK consists of g, $\mathcal{N}$ and the value $\theta = L(g^{m\beta}) = \alpha m\beta \ mod \ \mathcal{N}$, where the L-function takes in input elements from the set $S_{\mathcal{N}} = \{u < \mathcal{N}^2 | u = 1 \ mod \ \mathcal{N}\}$ and computes $L(u) = \frac{u-1}{\mathcal{N}}$.

  The secret key $sk = \beta \times m$ is shared with the Shamir scheme ([85]) modulo $m\mathcal{N}$. Let v be a square that generates with overwhelming probability the cyclic group of squares in $\mathbb{Z}^*_{\mathcal{N}^2}$. The verification keys vk_i are obtained with the formula $v^{\Delta sk_i} \ mod \ \mathcal{N}^2$.

- **Encryption** : To encrypt a message M, randomly pick $x \in_R \mathbb{Z}_{\mathcal{N}}$ and compute $c = g^M x^{\mathcal{N}} \ mod \ \mathcal{N}^2$.

- **Partial Decryption** : The i-th player P_i computes the decryption share $c_i = c^{2\Delta sk_i} \ mod \ \mathcal{N}^2$ using his secret share sk_i. He makes a proof of correct decryption which assures that $c^{4\Delta} \ mod \ \mathcal{N}^2$ and $v^{\Delta} \ mod \ \mathcal{N}^2$ have been raised to the same power sk_i in order to obtain c_i^2 and vk_i.

- **Combining Decryption** : If less than t decryption shares have valid proofs of correctness the algorithm fails. Otherwise, let S be a set of $t + 1$ valid shares and compute the plaintext

$$M = L\Big(\prod_{j \in S} c_j^{2\mu^S_{0,j}} \ mod \ \mathcal{N}^2\Big) \times \frac{1}{4\Delta^2\theta} \ mod \ \mathcal{N}$$

  where $\mu^S_{0,j} = \Delta \times \prod_{j' \in S \setminus \{j\}} \frac{j'}{j'-j}$.

3.4.2 Calculations on encrypted polynomials

In our protocol, we need do some calculations on encrypted polynomials. For a polynomial $f(x) = \sum_{i=0}^{m} a_i x^i$, we use $E(f(x))$ to denote the sequence of encrypted coefficients $\{E(a_i) | i = 0, ..., m\}$. Given $E(f(x))$, where $E(\cdot)$ is an additive HE scheme (e.g., Paillier), some computations can be made as follows (which have also been used in [36] and [62]):

1) At a value v, we can evaluate $E(f(x))$:

$$\begin{aligned} E(f(v)) &= E(a_m v^m + a_{m-1} v^{m-1} + ... + a_0) \\ &= E(a_m)^{v^m} E(a_{m-1})^{v^{m-1}} \cdots E(a_0). \end{aligned}$$

2) Given $E(f(x))$, we can compute the product $E(c \cdot f(x)) = \{E(a_m)^c, ..., E(a_0)^c\}$.

3) Given $E(f(x))$ and $E(g(x))$, $g(x) = \sum_{j=0}^{m} b_j x^j$, we can compute the sum:

$$E(f(x) + g(x)) = \{E(a_m)E(b_m), ..., E(a_0)E(b_0)\}.$$

4) Given $f(x)$ and $E(g(x))$, we can compute $E(f(x) * g(x))$. Suppose that $g(x) = \sum_{j=0}^{n} b_j x^j$, $f(x) * g(x) = \sum_{k=0}^{m+n} c_k x^k$, then

$$E(c_k) = E(a_0 b_k + a_1 b_{k-1} + ... + a_k b_0) = E(b_k)^{a_0} \cdots E(b_0)^{a_k}.$$

a_i or b_j are treated as zero if $i > m$ or $j > n$.

3.4.3 Notations

The major notations in this chapter are listed in Table 3.1.

Table 3.1: Major Notations in This Chapter

Notation	*Definition*
N	Total number of parties
P_i	The i-th party
T_i	The set or multiset on P_i
S	Total number of elements on each party
$T(i,j)$	The j-th element on P_i, $j = 1, ..., S$
c	Total number of colluded parties, $1 \leq c \leq N-1$
I	The index set of c colluded parties, $\{i_1, ..., i_c\}$
I'	The index set of honest parties, $\{1, ..., N\} \setminus I$
f_i	The polynomial whose roots are elements in T_i. $f_i = \prod_{j=1}^{S}(x - T(i,j))$
$\mathbb{Z}_{\mathcal{N}}$	The plaintext space of Paillier's cryptosystem

3.5 Protocol for Privacy Preserving Set Intersection in the semi-honest model

3.5.1 Main Idea

Constructing the Polynomial Vector F Our protocol for PPSI is based on evaluating randomized polynomials representing the intersection, which is a similar way with [36] and [62], but achieves lower cost.

Each P_i can compute a polynomial f_i to represent its set T_i:

$$f_i = (x - T(i,1)) \cdots (x - T(i,S)) \ mod \ \mathcal{N}$$

Then it randomizes f_i to be $f_i * \sum_{j=1}^{N} r_{i,j}$ by the help of other parties, in which $r_{i,j}$ is generated by P_j, $r_{i,j} = a_{i,j}x + b_{i,j}$, $a_{i,j}$ and $b_{i,j}$ are uniformly selected from the plaintext space of the threshold HE scheme (for Paillier's scheme, it is $\mathbb{Z}_{\mathcal{N}}$).

The N parties get a polynomial vector:

$$F = (f_1 * \sum_{j=1}^{N} r_{1,j}, ..., f_N * \sum_{j=1}^{N} r_{N,j})$$

Multiplication with Nonsingular Matrices Then the N parties compute $G = F \cdot R$, in which R is an $N \times N$ nonsingular matrix whose entries R_{uv} $(1 \leq u, v \leq N)$ are random numbers. The resulting G is another polynomial vector $(g_1, ..., g_N)$ as following:

$$\begin{aligned} g_1 &= f_1 * \sum_{j=1}^{N} r_{1,j} R_{11} + ... + f_N * \sum_{j=1}^{N} r_{N,j} R_{N1} \\ &... \\ g_N &= f_1 * \sum_{j=1}^{N} r_{1,j} R_{1N} + ... + f_N * \sum_{j=1}^{N} r_{N,j} R_{NN} \end{aligned} \tag{3.3}$$

Then, each P_i evaluates $(g_1, ..., g_N)$ at the element $T(i, j)$. If for $k = 1, ..., N$ $g_k(T(i, j)) = 0$, then P_i determines $T(i, j) \in TI$. The correctness of this determination will be proved in Lemma 6.

In the computation of G, to protect the privacy of each f_i, f_i is encrypted by P_i, and the encryption of $f_i * \sum_{j=1}^{N} r_{i,j}$ is computed. Then each party P_i generates a random matrix R_i so that $R = \prod_{i=1}^{N} R_i$ is nonsingular but no one knows what R is without publishing all R_i. The encryptions of FR_1, FR_1R_2, ..., $FR_1 \cdots R_N$ are computed respectively on P_1, P_2, ..., P_N. Finally, the N parties get the encryption of $G = FR$. After decryption, each P_i knows G, but not $f_{i'}$ for $i' \neq i$.

On each P_i, R_i should be generated as a nonsingular matrix with each entry is a random number over $\mathbb{Z}_{\mathcal{N}}$. In the appendix 3.10, we give the method to generate such an R_i on each P_i, which is from [79].

3.5.2 The Protocol

Protocol 1: *Protocol for Privacy Preserving Set Intersection in the semi-honest model*

Inputs: There are N $(N \geq 2)$ parties, any of which may be semi-honest. Each party has a private set of S elements, denoted T_i. Each party holds the public key and it is own share of the secret key for the threshold Paillier's cryptosystem.

Output: Each party P_i knows $TI = T_1 \cap ... \cap T_N$.

1) Computing $E(F)$: For $i = 1, ..., N$,

1.1) P_i computes $f_i = (x - T(i, 1)) \cdots (x - T(i, S))$, encrypts the coefficients to get $E(f_i)$, and sends $E(f_i)$ to all the other $N - 1$ parties.

1.2) on each P_j $(j \neq i)$, $r_{i,j}$ is generated as $a_{i,j}x + b_{i,j}$, in which $a_{i,j}$ and $b_{i,j}$ are uniformly selected from $\mathbb{Z}_{\mathcal{N}}$. P_j computes $E(f_i * r_{i,j})$ by computation 4) in Section 3.4.2, and sends it to P_i.

1.3) P_i also generates $r_{i,i}$ and computes $E(f_i * r_{i,i})$. Then P_i computes $E(f_i * \sum_{j=1}^{N} r_{i,j})$ by computation 3) in Section 3.4.2, and sends it to P_1.
In the end, P_1 gets $E(F)$ in which $F = (f_1 * \sum_{j=1}^{N} r_{1,j}, ..., f_N * \sum_{j=1}^{N} r_{N,j})$.

2) Computing $E(G)$: For $i = 1, ..., N$,

2.1) P_i generates a nonsingular $N \times N$ matrix R_i which is uniformly distributed over $\mathbb{Z}_N$ (by the method in [79]).

2.2) P_i computes $E(FR_1 \cdots R_i)$ according to computation 2) and 3) in Section 3.4.2, and sends it to P_{i+1} if $i + 1 \leq N$.
In the end, P_N gets $E(G) = E(F \prod_{i=1}^{N} R_i)$ and sends it to all the other parties.

3) Decryption and Evaluation :

3.1) The parties cooperatively decrypt $E(G)$ and gets $G = F(\prod_{i=1}^{N} R_i)$. Let $R = \prod_{i=1}^{N} R_i$, and $R_{u,v}$ $(1 \leq u, v \leq N)$ is the (u, v)-th entry of R, G is a polynomial vector $(g_1, ..., g_N)$ as described in equation 3.3) of Section 3.5.1.

3.2) Every P_i evaluates $T(i, j)$ in G for $j = 1, ..., S$ by computation 1) in Section 3.4.2. If $G(T(i, j)) = (\, g_1(T(i, j)), ..., g_N(T(i, j))\,) = (0, ..., 0)$, the $T(i, j) \in TI$; otherwise, $T(i, j) \notin TI$.

We prove the correctness of Protocol 1 in the following lemma:

Lemma 6 *Protocol 1 is a correct protocol for the PPSI problem.*

Proof: Protocol 1 determines whether $T(i, j) \in TI$ by $G(T(i, j))$. If $T(i, j) \in TI$, $T(i, j)$ is a root of all f_i for $i = 1, ..., N$, then

$$\begin{aligned} F(T(i,j)) &= (f_1(T(i,j)) \sum_{j=1}^{N} r_{1,j}, ..., f_N(T(i,j)) \sum_{j=1}^{N} r_{N,j}) \\ &= (0, ..., 0), \\ G(T(i,j)) &= F(T(i,j))R = (0, ..., 0). \end{aligned}$$

That is, if the evaluation $G(T(i, j)) \neq (0, ..., 0)$, $T(i, j) \notin TI$.
Then we prove that if $G(T(i, j)) = (0, ..., 0)$, overwhelmingly $T(i, j) \in TI$.
$G = FR_1 \cdots R_N = F(\prod_{i=1}^{N} R_i) = FR$. Because R_i for $i = 1, ..., N$ are generated to be nonsingular, $R = \prod_{i=1}^{N} R_i$ is also nonsingular. If $G(T(i, j)) = (0, ..., 0)$, a linear system $F(T(i, j))R = (0, ..., 0)$ can be made, and it has only one solution: $F(T(i, j)) = (0, ..., 0)$, i.e.,

$$f_l(T(i,j)) * \sum_{j=1}^{N} r_{l,j}(T(i,j)) = 0 \ for \ l = 1, ..., N.$$

The coefficients of $r_{l,j}$ are uniformly selected from $\mathbb{Z}_{\mathcal{N}}$. Suppose $\sum_{j=1}^{N} r_{l,j} = a_l x + b_l$, a_l and b_l are also uniformly distributed over $\mathbb{Z}_{\mathcal{N}}$. The probability that any $T(i, j) \in \mathbb{Z}_{\mathcal{N}}$

is a root of $a_l x + b_l$ is $1/\mathcal{N}$. If $\exists T(i,j), \forall l \in \{1,...,N\}$, $f_l(T(i,j)) * \sum_{j=1}^{N} r_{l,j}(T(i,j)) = 0$, because $f_l(T(i,j))$ must be 0 when $l = i$, so the probability that $\forall l\ (l \neq i)\ f_l(T(i,j)) = 0$ is $p = (1 - 1/\mathcal{N})^{N-1}$. N is the number of parties and practically $N \ll \mathcal{N}$. When $\mathcal{N}$ is large enough, $p \to 1$, then overwhelmingly $T(i,j)$ is a root of all f_l and $T(i,j) \in TI$. ■

3.6 Protocol for Privacy Preserving Set Intersection in the malicious model

3.6.1 Zero-Knowledge Proofs

Efficiently we can construct the following proofs based on proofs of knowledge on statements about discrete logarithms ([15]), the completeness and soundness of which have been argued respectively in the related work. Our constructions compose the basic proofs using AND ($\wedge$) operations, the closure of which is also argued in [18]. These proofs are used in our protocol for the malicious model.

1) Proof of knowing the plaintexts of the encrypted coefficients, $PK\{f : E(f)\}$. $E(f)$ are encrypted coefficients of the polynomial f. This is from the proof of knowing a plaintext in [15]. We give the basic proof in the Appendix 3.10.3.

2) Proof of correct polynomials multiplication, $PK\{r : \mathcal{M} = E(f * r) \bigwedge E(f) \bigwedge E(r)\}$. $f * r$ is the multiplication of polynomials f and r. This is from the proof of correct multiplications in [15]. We also give the basic proof in the Appendix 3.10.3.

3) Proof of the nonsingularity of an encrypted matrix, $PK\{R : \mathcal{D} \neq E(0) \bigwedge \mathcal{D} = E(det(R)) \bigwedge \mathcal{R} = E(R)\}$. R is an $N \times N$ matrix, $E(R)$ are the encrypted entries of R, $det(R)$ is the determinant of R. This is from the proof of correct multiplications in [15] and plaintext equality test in the Appendix 3.10.3.

4) Proof of correct matrix multiplication, $PK\{R : \mathcal{G} = E(FR) \bigwedge \mathcal{F} = E(F) \bigwedge \mathcal{R} = E(R)\}$. $F = (f_1, ..., f_N)$ is a vector of polynomials in which the i-th entry is a polynomial f_i for $i = 1, ..., N$. R is an $N \times N$ matrix, and $E(R)$ are the encrypted entries of R. This is also from the proof of correct multiplications in [15].

3.6.2 Protocols for the Malicious Model

We assume the adversary controls arbitrary number of parties, i.e., suppose the number is c, then $1 \leq c \leq N - 1$. We improve Protocol 1 to be Protocol 2, preventing the following malicious behaviors of the adversary by the zero-knowledge proofs in Section 3.6.1:

1) arbitrarily sending to others an encrypted polynomial without knowing its coefficients, e.g., just sending to others the encrypted polynomial received from other parties. Therefore, each party P_i should prove that he knows the coefficients of the polynomial he encrypts or multiplies, with the proof 1) and 2) of Section 3.6.1.

2) encrypting a polynomial whose coefficients are all zeros, then in equation 3.3) the malicious adversary will know the intersection of all honest parties. In Protocol 2, the honest parties can reset the leading coefficient of polynomials received from

others to be $E(1)$, and then g_k for $k = 1, ..., N$ can still hide the polynomials of the honest parties.

3) generating a singular matrix R_i, then if $G(T(i,j)) = (0, ..., 0)$, it is unnecessary that all $f_l(T(i,j)) = 0$ for $l = 1, ..., N$. Therefore, each party P_i should prove that R_i it generates is nonsingular with the proof 3) of Section 3.6.1.

4) doing multiplication with a matrix R_i' other than the committed matrix R_i. Each party should prove that he does correct matrix multiplication with the matrix R_i it has committed, with the proof 4) of Section 3.6.1.

Under these zero-knowledge proofs, each party should either behave in a semi-honest manner or being detected as cheating.

Protocol 2: *Protocol for Privacy Preserving Set Intersection in the malicious model*

Inputs: There are N ($N \geq 2$) parties, any of which may be malicious. Each party has a private set of S elements, denoted T_i. Each party holds the public key and its own share of the secret key for the threshold HE cryptosystem.

Output: Each party P_i knows $TI = T_1 \cap ... \cap T_N$.

Steps:

1) Computing $E(F)$: For $i = 1, ..., N$,

1.1) P_i computes $f_i = (x - T(i,1)) \cdots (x - T(i,S))$, encrypts the coefficients to get $E(f_i)$, and sends $E(f_i)$ to all the other parties with the proof of knowing the plaintexts of the encrypted coefficients, $PK\{f_i : E(f_i)\}$, excluding the leading coefficient.

1.2) For $j = 1, ..., N$, each P_j sets the leading coefficient of $E(f_i)$ to be $E(1)$, generates a random polynomial $r_{i,j}$. P_j computes $\mathcal{M}_{i,j} = E(f_i * r_{i,j})$, $\mathcal{P}_{i,j} = E(r_{i,j})$, and sends them to all the other partes, with the proof of correct polynomials multiplication:

$$PK\{r_{i,j} : \mathcal{M}_{i,j} = E(f_i * r_{i,j}) \bigwedge E(f_i) \bigwedge \mathcal{P}_{i,j} = E(r_{i,j})\}.$$

1.3) All P_j for $j = 1, ..., N$ compute $E(f_i * \sum_{j=1}^{N} r_{i,j})$.

In the end, all P_i for $i = 1, ..., N$ get $E(F)$ in which $F = (f_1 * \sum_{j=1}^{N} r_{1,j}, ..., f_N * \sum_{j=1}^{N} r_{N,j})$.

2) Computing $E(G)$: For $i = 1, ..., N$,

2.1) P_i generates a random nonsingular $N \times N$ matrix R_i, and sends $\mathcal{R}_i = E(R_i)$ and $\mathcal{D}_i = E(det(R_i))$ to all the other parties, with the proof of the nonsingularity of the encrypted matrix:

$$PK\{R_i : \mathcal{D}_i \neq E(0) \bigwedge \mathcal{D}_i = E(det(R_i)) \bigwedge \mathcal{R}_i = E(R_i)\}.$$

2.2) P_i computes $\mathcal{G}_i = E(FR_1 \cdots R_i)$, and sends it to all the other parties, with the proof of correct matrix multiplication:

$$PK\{R_i : \mathcal{G}_i = E(FR_1 \cdots R_i) \bigwedge \mathcal{G}_{i-1} = E(FR_1 \cdots R_{i-1}) \bigwedge \mathcal{R}_i = E(R_i)\}.$$

In the end, all P_i for $i = 1, ..., N$ get $E(G) = E(F \prod_{i=1}^{N} R_i)$.

3) Decryption and Evaluation :

3.1) All parties cooperatively decrypt $E(G)$.

3.2) Every P_i evaluates $T(i, j)$ in G for $j = 1, ..., S$. If $G(T(i, j)) = (0, ..., 0)$, the $T(i, j) \in TI$; otherwise, $T(i, j) \notin TI$.

3.7 Security Analysis

3.7.1 Security Analysis on Protocol 1

The Inferred Information by the Definition of PPSI

Suppose there are c colluded parties P_I, $I = \{i_1, ..., i_c\}$. It is unavoidable for P_I to combine their inputs and outputs to infer information. However, by the definition of PPSI in Section 3.3, they should know no more information than TI in each $T_{i'}$, $\forall i' \in I'$, $I' = \{1, ..., N\} \setminus I$. That is,

1) On $P_i \in P_I$, if $T(i, j) \in TI$, they know each $T_{i'}$ has $T(i, j)$.

2) On $P_i \in P_I$, if $T(i, j) \notin TI$, they don't know whether $T(i, j) \in T_{i'}$ for $\forall i' \in I'$.

The Inferred Information after Participating in Protocol 1

In Protocol 1, each P_i gets $G = (g_1, ..., g_N)$, so P_I may infer the roots of $f_{i'}$ for $\forall i' \in I'$ by analyzing the coefficients in G. By the following lemma, we prove that G resists such kind of analysis.

Lemma 7 *In Protocol 1, any P_i in the coalition of c ($1 \leq c \leq N-1$) semi-honest parties (P_I) can know no more elements than TI in any $T_{i'}$ for $\forall i' \in I'$.*

Proof: Due to the security of the threshold HE cryptosystem, P_I can't know any information on the plaintexts of the encryptions unless they are decrypted. We use P_i to denote any party in P_I. P_i gets only the decryption of $E(G)$. If $G(T(i, j)) = (0, ..., 0)$, by Lemma 6, P_i knows $T(i, j)$ is a root for all f_l ($l = 1, ..., N$) and each $T_{i'}$ has $T(i, j)$. This accords with the case 1) in "The Inferred Information by the Definition of PPSI".

1) We firstly prove that, if $G(T(i, j)) \neq (0, ..., 0)$, P_i doesn't know whether $T(i, j) \in T_{i'}$ for $\forall i' \in I'$, that is, whether $T(i, j)$ is a root of any $f_{i'}$.

From the view of P_i,

$$G = F(\prod_{i\in I} R_i \cdot \prod_{i'\in I'} R_{i'}),\ in\ which$$
$$\prod_{i\in I} R_i\ is\ generated\ by\ P_I,$$
$$\prod_{i'\in I'} R_{i'}\ is\ generated\ by\ P_{I'}.$$

P_i doesn't know $\prod_{i'\in I'} R_{i'}$, thus if $G(T(i,j)) \neq (0,...,0)$, P_i can't compute $F(T(i,j))$. Then P_i can't know any $f_{i'}(T(i,j))$ and whether $T(i,j) \in T_{i'}$ for $\forall i' \in I'$. This accords with the case 2) in "The Inferred Information by the Definition of PPSI".

2) P_i may also analyze the coefficients of a single g_l $(l = 1,...,N)$. From P_i's view,

$$g_l = f_{TI}(\mathcal{F}_I + \mathcal{F}_{I'}),\ in\ which$$
$$f_{TI}\ is\ the\ polynomial\ whose\ roots\ are\ TI,$$
$$\mathcal{F}_I = \sum_{i\in I}(f_i/f_{TI} * \sum_{j=1}^{N} r_{i,j}R_{il}),$$
$$\mathcal{F}_{I'} = \sum_{i'\in I'}(f_{i'}/f_{TI} * \sum_{j=1}^{N} r_{i',j}R_{i'l}).$$

We should also prove that P_i can't know $\mathcal{F}_{I'}$, otherwise he will know $\bigcap_{i'\in I'} T_{i'}$ by factoring $\mathcal{F}_{I'}$.

From the view of P_i, in $\mathcal{F}_I$, $\forall i \in I$, $\sum_{j=1}^{N} r_{i,j}R_{il}$ can be supposed as $b_{i,1}x + b_{i,0}$, in which $b_{i,1}$ and $b_{i,0}$ are random numbers. Given $f_i/f_{TI} = \sum_{k=0}^{S-|TI|} a_{i,k}x^k$, suppose $f_i/f_{TI} * \sum_{j=1}^{N} r_{i,j}R_{il} = \sum_{k=0}^{S-|TI|+1} c_{i,k}x^k$, then $c_{i,k} = a_{i,k}b_{i,0} + a_{i,k-1}b_{i,1}$.

Suppose $\mathcal{F}_I = \sum_{k=0}^{S-|TI|+1} e_k x^k$, then $e_k = \sum_{i\in I} c_{i,k}$.

Suppose $\mathcal{F}_{I'} = \sum_{k=0}^{S-|TI|+1} e'_k x^k$, then the k-th coefficient of $\mathcal{F}_I + \mathcal{F}_{I'}$: $e''_k = e_k + e'_k = \sum_{i\in I}(a_{i,k}b_{i,0} + a_{i,k-1}b_{i,1}) + e'_k$.

P_i knows all e''_k from g_l/f_{TI}, and all $a_{i,k}$ from f_i/f_{TI}, but doesn't know all $b_{i,1}$, $b_{i,0}$, and e'_k. Thus from $e''_k = \sum_{i\in I}(a_{i,k}b_{i,0}+a_{i,k-1}b_{i,1})+e'_k$, P_i gets a set of $S-|TI|+2$ linear equations with $2c+S-|TI|+2$ unknowns. For $1 \leq c \leq N-1$, P_i can't compute the solutions for these unknowns. Therefore, P_i can't know e'_k for $k = 0,...,S-|TI|+1$, and can't know any root of $\mathcal{F}_{I'}$.

In each g_l $(l = 1,...,N)$, P_i can't know $\mathcal{F}_{I'}$, which makes P_i fail to know any $f_{i'}/f_{TI}$ in $\mathcal{F}_{I'}$.

In sum, in Protocol 1, $P_i \in P_I$ can know no more roots than TI in any $T_{i'}$ for $\forall i' \in I'$. ■

Theorem 5 *Protocol 1 is a secure protocol* Π *in Definition 6, which privately solves the PPSI problem with respect to the semi-honest behaviors of arbitrary number of parties.*

The proof of this theorem is postponed in the Appendix 3.10.

3.7.2 Security Analysis on Protocol 2

Theorem 6 *Assuming the threshold Paillier encryption is semantically secure and the zero-knowledge proofs in Protocol 2 can't be forged, Protocol 2 is a secure protocol* Π *in Definition 7, which securely solves the problem of PPSI when the number of malicious parties is arbitrary.*

This theorem can be proved following the ideal-vs.-real emulation paradigm. The detailed proof is postponed to the Appendix 3.10.

3.8 Comparisons with Related Work

3.8.1 Comparisons for Protocol 1

−Complexity of Protocol 1 :

1) *Computation Cost*: Each Paillier's encryption and decryption requires a cost of $2lg\mathcal{N}$ modular multiplications (mod $\mathcal{N}^2$). Each exponentiation has the same cost with the encryption. We compare our protocol with other related work regarding their computation cost on encryptions and multiplications of ciphertexts, and consider modular multiplication (mod $\mathcal{N}^2$) as a basic computation.

   Thus, for each party in Protocol 1, the total encryptions are $(S+2)(N-1)^2-2$, and the total multiplications of ciphertexts are $(S+2)(N^2+2N-3)$. Then the total computation cost for each party is $2((S+2)(N-1)^2-2)lg\mathcal{N}+(S+2)(N^2+2N-3)$ modular multiplications.

2) *Communication Cost*: The length of each encryption is $2lg\mathcal{N}$. Then in Protocol 1, the total communication cost among all parties is $2N(N-1)(4S+5)lg\mathcal{N}$ bits.

Speeding up techniques can be employed in Protocol 1. If all parties ensure that there is a coalition of c ($1 \leq c \leq N-1$) semi-honest parties, in Step 1) of Protocol 1 each $E(f_i)$ can be randomized as $E(f_i * \sum_{j=1}^{c+1} r_{i,j})$ by sending $E(f_i)$ to any c parties, instead of all the other $N-1$ parties. In Step 2) $E(G)$ can be computed as $E(F\prod_{i=1}^{c+1} R_i)$. What's more, in Step 1) the iterations $i = 1, ..., N$ can be made in parallel. Then the computation cost is $2(c(S+2)(N-1)-2)lg\mathcal{N}+c(S+2)(N+3)$ modular multiplications. The communication cost is $2cN(4S+5)lg\mathcal{N}$ bits.

−Kissner's Protocol : In Kissner's protocol for PPSI ([62]), a single polynomial F is constructed and evaluated on each $T(i,j)$:

$$F = \sum_{l=1}^{N} f_l * \sum_{k=1}^{N} r_{l,k}.$$

f_l is a polynomial representing elements on P_l, $r_{l,k}$ is a polynomial uniformly selected by P_k and has the same degree with f_l. In this protocol, it is easy to see that $T(i,j) \in TI$ is a sufficient condition for the evaluation $F(T(i,j)) = 0$, but $F(T(i,j)) = 0$ is not a sufficient condition for $\forall l \in \{1, ..., N\}$, $f_l(T(i,j)) * \sum_{k=1}^{N} r_{l,k}(T(i,j)) = 0$.

In Lemma 6 we have proved that if $\forall l \in \{1, ..., N\}$, $f_l(T(i,j)) * \sum_{k=1}^{N} r_{l,k}(T(i,j)) = 0$, the probability that $T(i,j) \in TI$ is $(1-1/\mathcal{N})^{N-1}$. Therefore, in Kissner's protocol, if

$F(T(i,j)) = 0$, the probability that $T(i,j) \in TI$ is less than the probability achieved by our Protocol 1.

The major cost of this protocol is on computing the encrypted F. It is also based on Paillier's cryptosystem. The computation cost for each party and communication cost among all parties are shown in Table 3.2.

−**Freedman's Protocol** : In Freedman's protocol for PPSI ([36]), each party P_i ($i = 1, ..., N-1$) sends the encrypted polynomial f_i representing its elements to P_N. P_N evaluates its elements $T(N,j)$ for $j = 1, ..., S$ on all these polynomials, randomizes the evaluations and sends them to all the other parties. These parties decrypt and combine the evaluations to determine whether $T(i,j) \in TI$. In this protocol each party also generates a random matrix, but the matrices are used in a different way from our Protocol 1 for they aren't full rank and not for multiplications. The XOR of each row of the matrices is required to be zero, and they are used to randomize the decryptions on each party. The major cost of this protocol is on the evaluations of encrypted polynomials at all elements of P_N. The protocol is also based on Paillier's cryptosystem. The average computation cost for each party and communication cost among all parties are shown in Table 3.2. In [36] only the protocol for the semi-honest model is given.

−**Comparisons of 3 protocols** : From Table 3.2, the computation costs of Protocol 1, protocol in [62] and [36] are respectively $O(cSNlg\mathcal{N})$, $O(cS^2lg\mathcal{N})$, $O(S(S+N)lg\mathcal{N})$. Practically the size of a set, S, may be much larger than the number of parties, N. Then it is easy to see that Protocol 1 is more efficient in computation than [62] and [36], and more efficient in communication than [62].

For a quantitative analysis, we conservatively set $S = 20$, $N = 5$, and set $c = 3$, $lg\mathcal{N} = 1024$, then Protocol 1 saves 81% and 63% computation costs, 17% and 20% communication costs in comparison with [62] and [36]. We also notice that if $c = 4$, i.e., all of the N parties are semi-honest, then the communication cost in Protocol 1 will increase by 6% in comparison with [36]. Thus Protocol 1 can utilize the knowledge on honest relationships among some of the N parties to reduce the communication cost.

Table 3.2: Comparisons of solutions for the PPSI problem in the semi-honest model

	Computation Cost	*Communication Cost*	*Security Model*
Our Protocol 1	$2(c(S+2)(N-1)-2)lg\mathcal{N}$ $+c(S+2)(N+3)$	$2cN(4S+5)lg\mathcal{N}$	Semi-honest
Protocol in [62]	$2(c(S+1)^2+5S+3)lg\mathcal{N}$ $+c(S^2+4S+2)$	$2cN(5S+2)lg\mathcal{N}$	Semi-honest
Protocol in [36]	$((S+1)(S+2)+3S(N-1)-1)2lg\mathcal{N}$ $+S(S+1)$	$10S(N-1)^2lg\mathcal{N}$	Semi-honest

3.8.2 Comparisons for Protocol 2

The complexity of Protocol 2 is determined by the complexity of Protocol 1, and a linear combination of the complexity of the zero-knowledge proofs in Section 3.6.1. The proofs in Section 3.6.1 are based on the basic blocks, such as proof of knowing the plaintext, proof of correct multiplication, and private equality test, all of which have a computation cost of $O(lg\mathcal{N})$ modular multiplications, and a communication cost of $O(lg\mathcal{N})$ bits, according

to [15] and [58]. Thus Protocol 2 keeps the same level of complexity as Protocol 1, that is, it has a computation cost of $O(cSNlg\mathcal{N})$ modular multiplications, and a communication cost of $O(cSNlg\mathcal{N})$ bits.

In [63] the PPSI protocol for the semi-honest model is also extended to the malicious model. The computation cost of that protocol is $O(cS^2lg\mathcal{N})$ modular multiplications, and the communication cost is $O(cSNlg\mathcal{N})$ bits.

In the following Table 3.3, we list the costs of the two protocols. In practical applications, the size of a set S can be much larger than the number of parties N, so our Protocol 2 can be faster than the protocol in [63].

Table 3.3: Comparisons of solutions for the PPSI problem in the malicious model

	Computation Cost	*Communication Cost*	*Security Model*
Our Protocol 2	$O(cSNlg\mathcal{N})$	$O(cSNlg\mathcal{N})$	Malicious
Protocol in [63]	$O(cS^2lg\mathcal{N})$	$O(cSNlg\mathcal{N})$	Malicious

3.9 Concluding Remarks

We address the problem of Privacy Preserving Set Intersection (PPSI) among N parties. The problem is solved by constructing polynomials representing elements in the set intersection, and evaluating the polynomials to determine whether an element is in the set intersection, without publishing the datasets on each party. Our protocol is firstly constructed and the security is analyzed assuming there is a coalition of arbitrary c ($1 \leq c \leq N-1$) semi-honest parties. Then the protocol is extended by some zero-knowledge proofs and the security is also analyzed assuming there is a coalition of arbitrary number of malicious parties. In comparison with related work in [62], [36] and [63], our protocol has less computation and communication costs in both models.

3.10 Appendix

3.10.1 Proofs of Theorems

Theorem 1: *Protocol 1 is a secure protocol* Π *in Definition 6, which privately solves the PPSI problem with respect to the semi-honest behaviors of arbitrary number of parties.*

Proof: Protocol 1 provides a Π to compute $\mathfrak{f}$. Given any coalition of c ($c \leq N-1$) semi-honest parties indexed by $I = \{i_1, ..., i_c\}$, by Definition 6 in Section 3.3, we have to show that there exists a PPT algorithm $\mathcal{S}$ such that $\mathcal{S}(I, (T_{i_1}, ..., T_{i_c}), \mathfrak{f}_I(\overline{T}))$ and $VIEW_I^{\Pi}(\overline{T})$ are computationally indistinguishable.

$VIEW_I^{\Pi}(\overline{T}) = \{V_1, V_2, V_3, V_4\}$:

- V_1 is $I = \{i_1, ..., i_c\}$.
- V_2 are $T_{i_1}, ..., T_{i_c}$.
- V_3 are $E(G)$ and the intermediate encryptions received by P_I.

- V_4 are $G(T(i_t, j))$ for any $i_t \in I$.

With these views, the coalition can do the following two types of analysis:

1) *Cryptanalysis on* (V_1, V_2, V_3): Due to the semantic security of the threshold HE cryptosystem, P_i can't gain extra information from the encryptions in V_3. That is, supposing V_3 has s encryptions, with only negligible probability, P_i can distinguish V_3 and $\mathcal{ER}_1 = (E(r_1), ..., E(r_s))$ by randomly choosing $\mathcal{R}_1 = (r_1, ..., r_s)$ over the plaintext space of the HE scheme. Thus, $(V_1, V_2, V_3) \equiv^c (V_1, V_2, \mathcal{R}_1, \mathcal{ER}_1)$.

2) *Roots analysis on* (V_1, V_2, V_4): From Lemma 7, P_I can't know roots other than TI in any $f_{i'}$ for $\forall i' \in I'$. Thus, $V_4 = (\mathcal{A}, TI)$. $\mathcal{A} = \{a_{i_t,j} | i_t \in \{i_1, ..., i_c\}, j = 1, ..., S\}$ in which $a_{i_t,j} = 1$ if $G(T(i_t, j)) = (0, ..., 0)$, and $a_{i_t,j} = 0$ otherwise.

In sum, $VIEW_I^{\Pi}(\overline{T}) \equiv^c (V_1, V_2, \mathcal{R}_1, \mathcal{ER}_1, \mathcal{A}, TI)$.

$\mathfrak{f}_I(\overline{T}) = (\mathcal{A}, TI)$ by the analysis in Section 3.7.1. Let $\mathcal{R}'_1 = \{r'_i | i = 1, ..., s\}$ are randomly chosen by P_I, and $\mathcal{ER}'_1$ are the encryptions of the sequence in $\mathcal{R}'_1$, then

$$\begin{aligned} \mathcal{S}(I, (T_{i_1}, ..., T_{i_c}), \mathfrak{f}_I(\overline{T})) &\equiv^c (I, (T_{i_1}, ..., T_{i_c}), \mathcal{A}, TI, \mathcal{R}'_1, \mathcal{ER}'_1) \\ &\equiv^c (V_1, V_2, \mathcal{A}, TI, \mathcal{R}_1, \mathcal{ER}_1) \\ &\equiv^c VIEW_I^{\Pi}(\overline{T}). \end{aligned}$$

Then Protocol 1 privately computes PPSI against the coalition of any c ($c \leq N - 1$) semi-honest parties. ■

Theorem 2: *Assuming the threshold Paillier encryption is semantically secure and the zero-knowledge proofs in Protocol 2 can't be forged, Protocol 2 is a secure protocol* Π *in Definition 7, which securely solves the problem of PPSI when the number of malicious parties is arbitrary.*

Proof: Suppose A and B are respectively adversarial strategies in the real and ideal model, and they control the same set of parties P_I ($1 \leq |I| \leq N-1$) during the executions. Protocol 2 actually provides a Π to compute the function $\mathfrak{f}$ as defined in Section 3.3. We need to prove that the views of A and B are computationally indistinguishable, in order to prove the the joint executions $\{IDEAL_{\mathfrak{f},I,B}(\overline{T})\} \equiv^c \{REAL_{\Pi,I,A}(\overline{T})\}$.

Firstly we analyze the view of A. In the real execution of Protocol 2, A can't know the plaintexts of encryptions received from the honest parties, can't extract information other than the statements in the zero-knowledge proofs in Step 1.1), 1.2), 2.1), 2.2), and can't convince the honest parties on any false statement. In Step 1.2), A can't commit zero polynomials so that in Step 3.1), $\forall k \in \{1, ..., N\}$, g_k won't become $\sum_{l \in I'} f_l * \sum_{j=1}^{N} r_{l,j} R_{l,k}$ (I' is the set of honest parties), otherwise A will know the intersection of $\bigcap_{i \in I'} P_i$. By Lemma 7, A can't know more elements than TI on P_i ($i \in I'$).

Secondly we analyze the view of B. In the ideal model, the honest parties (denoted by $P_{I'}$) and malicious parties controlled by B compute $\mathfrak{f}$ by the help of the trusted third party (TTP). B can be constructed using A as a subroutine as following:

1) *Computing* F :

1.1) B invokes A. A intentionally generates f_j for each party P_j in it, and sends $E(f_j)$ to B. B sets the leading coefficient of $E(f_j)$ to be $E(1)$, and emulates the proof $PK\{f_j : E(f_j)\}$ to check whether a verifier will be convinced that P_j knows the plaintexts of each coefficient in $E(f_j)$ excluding the leading coefficient. If the verifier would be convinced, B sends f_j to the TTP, otherwise he aborts.

For the honest parties in $P_{I'}$, they send their polynomials directly to the TTP.

1.2) The TTP encrypts all f_i for $i = 1, ..., N$ and sends them to the honest parties and B. B invokes A again. B forwards all $E(f_i)$ to A. For $i = 1, ..., N$, A generates a random polynomial $r_{i,j}$ for each party P_j in it, computes $\mathcal{M}_{i,j} = E(f_i * r_{i,j})$, $\mathcal{P}_{i,j} = E(r_{i,j})$, and sends them to B. B checks whether a verifier will be convinced that the polynomials multiplication is correct by emulating the proof $PK\{r_{i,j} : \mathcal{M}_{i,j} = E(f_i * r_{i,j}) \bigwedge E(f_i) \bigwedge \mathcal{P}_{i,j} = E(r_{i,j})\}$. If the verifier would be convinced, B sends all $f_i * r_{i,j}$ to the TTP, otherwise he aborts.

For the honest parties P_j in $P_{I'}$, they generate a random polynomial $r_{i,j}$ for $i = 1, ..., N$ and send all $f_i * r_{i,j}$ directly to the TTP.

1.3) The TTP computes $F = (f_1 * \sum_{j=1}^{N} r_{1,j}, ..., f_N * \sum_{j=1}^{N} r_{N,j})$.

2) *Computing* $\mathcal{G}$: The TTP sets R_0 to be an $N \times N$ identity matrix. For $j = 1, ..., N$, the TTP computes $\mathcal{G}_j = FR_0 \cdots R_{j-1}$, and sends $E(\mathcal{G}_j)$ to P_j.

2.1) If P_j is in B, B invokes A. A generates a random nonsingular $N \times N$ matrix R_j for each P_j, and sends $\mathcal{R}_i = E(R_j)$ and $\mathcal{D}_j = E(det(R_j))$ to B. B checks whether a verifier will be convinced that R_j is nonsingular by emulating the proof of $PK\{R_j : \mathcal{D}_j \neq E(0) \bigwedge \mathcal{D}_j = E(det(R_j)) \bigwedge \mathcal{R}_j = E(R_j)\}$.
Then A computes $\mathcal{G}_{j+1} = E(FR_0 \cdots R_j)$, and sends it to B. B checks whether a verifier will be convinced that the matrix multiplication is correct by emulating the proof $PK\{R_j : \mathcal{G}_{j+1} = E(FR_0 \cdots R_j) \bigwedge \mathcal{G}_j = E(FR_0 \cdots R_{j-1}) \bigwedge \mathcal{R}_j = E(R_j)\}$. If the verifier would be convinced by both proofs, B sends R_j to the TTP, otherwise he aborts.

2.2) If P_j is an honest party, he generates a nonsingular matrix R_j and send it directly to the TTP.

3) *Evaluation* :

3.1) The TTP gets $G = FR_0 \cdots R_N$ and sends it to all parties.

3.2) Every P_i evaluates $T(i, j)$ in G for $j = 1, ..., S$. If $G(T(i, j)) = (0, ..., 0)$, the $T(i, j) \in TI$; otherwise, $T(i, j) \notin TI$.

According to the above procedure, in assumption of the Paillier encryption is semantically secure and the zero knowledge proofs can't be forged, the view of B is computationally indistinguishable from the view of A, so the joint executions $\{IDEAL_{\mathfrak{f},I,B}(\overline{T})\} \equiv^c \{REAL_{\Pi,I,A}(\overline{T})\}$. ■

3.10.2 Generation of Random and Nonsingular Matrix

In Protocol 1 and 2 we need each party to generate a random and nonsingular matrix over the plaintext space of Pailler's encryption scheme, $\mathbb{Z}_{\mathcal{N}}$. The generation algorithm is from [79], and to generate an $n \times n$ nonsingular matrix, the computation complexity is $O(n^2) + Mul(n)$, where $Mul(n)$ is the time needed to multiply two $n \times n$ matrices. Below we give the general ideas and some steps of the generation algorithm.

The algorithm simultaneously constructs two matrices whose product will be the nonsingular matrix we are looking for. It works recursively, fixing a row and column of each matrix at every stage. It is proved in [79] that the product of the two matrices will be nonsingular of full rank since the fixed row and column on the second matrix just acts as a linear transformation of some standard basis and this nonsingular minor.

The main ideas behind the algorithm are as follows. To generate a nonsingular matrix with first row $e_1 = (1, 0, 0, ..., 0)$ uniformly it suffices to choose the remaining elements in the first column uniformly from the field and choose the $(1, 1)^{th}$-minor uniformly from the set of $(n-1) \times (n-1)$ nonsingular matrices. We can generalize this to matrices of any fixed nonzero first row. Let $e_r =< 0, 0, ..., 1, ..., 0 >$, where the 1 is in the r^{th} row. Given a finite field F and a vector v in $F^n \setminus \{0^n\}$ with nonzero r^{th} coordinate, we describe a bijection between the set of nonsingular matrices with first row e_r and those with first row v. This bijection is the linear transformation achieved by multiplying on the right by a matrix T that is the identity matrix with the r^{th} row replaced with v. This provides a reduction from the problem of uniformly generating an $n \times n$ nonsingular matrix to that of generating an $(n-1) \times (n-1)$ nonsingular matrix and takes one matrix multiplication. The Algorithm is as follows:

Nonsing$(\mathbf{n})$:

begin

Gen$(\mathbf{A}, \mathbf{T}, \mathbf{n})$;
$return(A * T)$.

end

Gen$(\mathbf{A}, \mathbf{T}, \mathbf{n})$:

begin

if $n = 1$ then
let $A_{1,1} = 1$;
choose $T_{1,1} \in F \setminus \{0\}$;
else
choose $v \in F^n \setminus \{0^n\}$ such that $v \neq 0^n$;
let r be the first nonzero coordinate of v;
let the first row of $A = e_r$; /*Assign Values to A*/
let the r^{th} row of $T = v$;
for all $1 \leq i \leq n$, where $i \neq r$, $T_{i,r} = 0$; /*Assign Values to T*/
Gen$(\mathbf{A}[\mathbf{1}, \mathbf{r}], \mathbf{T}[\mathbf{r}, \mathbf{r}], \mathbf{n} - \mathbf{1})$

end

3.10.3 Some Basic Zero-knowledge Proofs

Proving Knowing a Plaintext In Paillier's cryptosystem, a prover can prove to the verifier that he knows a plaintext α in in the ciphertext $C_\alpha = g^\alpha s^{\mathcal{N}} \ mod \ \mathcal{N}^2$, by the following steps (from [15]):

1. P_i chooses $x \in \mathbb{Z}_{\mathcal{N}}$ and $u \in \mathbb{Z}^*_{\mathcal{N}^2}$ at random, computes and sends
$$A = g^x u^{\mathcal{N}} \ mod \ \mathcal{N}^2$$
2. The verifier sends a random challenge e.
3. P_i computes and sends
$$w = x + e\alpha \ mod \ \mathcal{N},$$
$$z = us^e g^t \ mod \ \mathcal{N}^2,$$
where t is defined by $x + e\alpha = w + t\mathcal{N}$.
4. The verifier checks that
$$g^w z^N = AC_\alpha^e \ mod \ \mathcal{N}^2$$
and accepts if and only if this is the case.

Proving Multiplications Correct We now describe a $\sum$-protocol for securely multiplying an encrypted value by a constant. So we have as input encryptions $C_a = g^a r^{\mathcal{N}} \ mod \ \mathcal{N}^2$, $C_\alpha = g^\alpha s^{\mathcal{N}} \ mod \ \mathcal{N}^2$, $D = C_a^\alpha \gamma^N \ mod \ \mathcal{N}^2$ and a player P_i knows in addition α, s, γ. What we need is a proof that D encrypts $a\alpha \ mod \ \mathcal{N}$. We proceed as follows:

1. P_i chooses $x \in \mathbb{Z}_{\mathcal{N}}$ and $v, u \in \mathbb{Z}^*_{\mathcal{N}^2}$ at random, computes and sends
$$A = C_a^x v^{\mathcal{N}} \ mod \ \mathcal{N}^2,$$
$$B = g^x u^{\mathcal{N}} \ mod \ \mathcal{N}^2$$
2. The verifier sends a random challenge e.
3. P_i computes and sends
$$w = x + e\alpha \ mod \ \mathcal{N},$$
$$z = us^e g^t \ mod \ \mathcal{N}^2,$$
$$y = vC_a^t \gamma^e \ mod \ \mathcal{N}^2$$
where t is defined by $x + e\alpha = w + t\mathcal{N}$.
4. The verifier checks that
$$g^w z^{\mathcal{N}} = BC_\alpha^e \ mod \ \mathcal{N}^2,$$
$$C_a^w y^{\mathcal{N}} = AD^e \ mod \ \mathcal{N}^2$$
and accepts if and only if this is the case.

Proving the Plaintext is Nonzero Our protocol 2 also needs a proof of nonzero plaintext, i.e., P_i needs to prove that it has encrypted a nonzero a in $C_a = g^a r^{\mathcal{N}} \ mod \ \mathcal{N}^2$. This proof can be based on the proof of correct multiplications above and the threshold cryptosystem. We proceed as follows:

1. P_i chooses a random $\alpha \in \mathbb{Z}_{\mathcal{N}}$ $(\alpha \neq 0)$, computes

$$C_\alpha = g^\alpha s^{\mathcal{N}} \ mod \ \mathcal{N}^2,$$
$$D = C_a^\alpha \gamma^{\mathcal{N}} \ mod \ \mathcal{N}^2.$$

2. P_i proves to the verifier that D encrypts $a\alpha \ mod \ \mathcal{N}$ by the proof of correct multiplication above.
3. P_i and the verifier cooperatively decrypt D using the threshold decryption keys. The verifier checks whether the decryption $a\alpha \ mod \ \mathcal{N}$ is zero. If it is nonzero, the verifier accepts that a is nonzero without knowing what a is.

Chapter 4

Privacy Preserving Tuple Matching in Distributed Database

4.1 Problem Background

Tuple matching is a basic problem that has been encountered in many applications of databases ([19]), such as tuple merge/purge, tuple deduplication, etc. The tuple merge/purge problem is the task of merging data from multiple sources in a manner as efficient as possible ([53]). Deduplication is to detect and eliminate duplicate records that refer to the same entity in spite of various data inconsistencies, when integrating data from multiple databases ([82]).

In distributed settings, privacy concern is an important issue when performing tuple matching, because data sources are reluctant to publish their tuples to others for their individual privacy. We focus this chapter on some simple tuple matching problems and address their privacy preservation issues. For example, supermarkets may ally to provide discounts for the following kinds of customers: 1) the customers who have purchased the same commodities in any two of the supermarkets, 2) the customers whose commodities in one certain purchase are all sold well. In both cases, the supermarkets would not like to publish their whole database to each other because of their own commercial secrecy. Therefore, there should be some privacy preserving techniques for a supermarket to determine whether its customers have duplicate purchase records in other supermarkets, or whether they have purchased a group of commodities which are all sold well in the alliance, without its database of customers published to other supermarkets. In this chapter, we name the former problem *privacy preserving tuple matching*, and the latter *privacy preserving threshold attributes matching*. We will show the two problems can be solved by similar techniques.

Problem Formulation : We are given a distributed database whose relation (table) is horizontally partitioned among different parties (physical locations), where each party $P_i (i = 1, ..., N,\ N \geq 2)$ has a set of tuples (records) $T_i = \{T(i,j) | j = 1, ..., S\}$ and $T(i,j) = \big(T(i,j)_1, ..., T(i,j)_M\big)$. All tuples have the same M attributes ($M \geq 2$).

We assume that each party has the same number of tuples. Practically if the numbers of tuples on all parties are different, they can use the maximum number as S, and the other parties can pad zero tuples in their databases to get S tuples. We also assume that the data inconsistencies inside attributes have been cleaned, then an attribute value $T(i,j)_k$ has a duplicate $T(i',j')_k$, if and only if $T(i,j)_k = T(i',j')_k$. In this way we also say that $T(i,j)_k$

Table 4.1: An example table

	A_1	A_2	A_3	A_4
T_1	a_1	b_1	c_1	d_1
T_2	a_2	b_1	c_2	d_2
T_3	a_1	b_1	c_1	d_1
T_4	a_2	b_1	c_2	d_3
T_5	a_1	b_1	c_1	d_1
T_6	a_1	b_1	c_2	d_4

has an *attribute match* with $T(i',j')_k$. $T(i,j)$ has a *tuple match* with $T(i',j')$, if and only if $T(i,j)$ has M attribute matches with $T(i',j')$, i.e., $\forall k \in \{1,...,M\}, T(i,j)_k = T(i',j')_k$.

1) *Privacy Preserving Tuple Matching (PPTM): Each party P_i determines whether each tuple $T(i,j)$ has any duplicate with the other parties, i.e., whether $T(i,j) \in T_i \cap (\bigcup_{i'=1...N, i' \neq i} T_{i'})$, without gaining any information other than those inferred from the combination of its inputs and determination outcomes.* PPTM can be used when each party cares about only the fact whether there is any duplicate with the other parties for each tuple of its own, but not the number of these duplicates.

2) *Privacy Preserving Threshold Attributes Matching (PPTAM): Each party P_i determines whether all attributes of $T(i,j)$, i.e., $\forall k \in \{1,...,M\}$, $T(i,j)_k$ appear at least 2 times in the k-th attribute's value union on the N parties, i.e., $A_k = \{T(i,j)_k | i = 1,...,N,\ j = 1,...,S\}$, without gaining any information other than those inferred from the combination of its inputs and determination outcomes.*

Actually, in PPTAM, each party P_i determines whether $T(i,j)$ is in $T1_i$ or $T0_i$: $T1_i$ is the set of tuples $\{T(i,j) | \forall k \in \{1,...,M\}, T(i,j)_k$ *appear at least 2 times in* $A_k\}$. $T0_i$ is the tuple set $\{T(i,j) | \exists M' \subseteq \{1,...,M\}, \forall k \in M', T(i,j)_k$ *appear only one time in* $A_k\}$. It is easy to see that $T1_i \cap T0_i = \phi$ and $T1_i \cup T0_i = T_i$.

As a simple example in Table 4.1, supermarket P_1 has customers T_1 and T_2, supermarket P_2 has T_3 and T_4, P_3 has T_5 and T_6. Every customer buys one type of each commodity A_i $(i = 1,2,3,4)$. When PPTAM is applied, P_1 finds $T_1 \in T1_1$ and $T_2 \in T0_1$; P_2 finds $T_3 \in T1_2$ and $T_4 \in T0_2$; P_3 finds $T_5 \in T1_3$ and $T_6 \in T0_3$. That is, the customers T_1, T_3 and T_5 have bought a group of types on (A_1, A_2, A_3, A_4), all of which have been sold for at least 2 times, thus they can be given some discounts by the supermarkets.

Privacy Requirements : We assume that there is only one adversary and he controls c $(c \leq N-1)$ parties, that is, there may be c colluded parties that combine their inputs and outcomes together to gain more information. Analyses on the information gained by this collusion are postponed to Section 4.6. If there is no collusion among the N parties, the *privacy requirements* from the definitions of PPTM and PPTAM still deserve to be pointed out as follows:

1) *In both problems, each party does not publish any tuple of its own.*

2) *In both problems, whatever the determination on $T(i,j)$ is, P_i does not know the specific matching times and the matching originations, i.e., which party $T(i,j)$ has a match with.*

3) *In the PPTAM problem, if the determination on* $T(i,j)$ *is* $T(i,j) \in T0_i$, P_i *does not know the attribute subset* M', *such that* $\forall k \in M'$, $T(i,j)_k$ *appears only one time in* A_k.

Our Contributions : Our main contributions in this chapter include:

1) Our proposed protocol for the PPTM problem has a lower computation cost than two related solutions in [62] and [63], and keeps the same security defined under Secure Multi-party Computation (SMC), without increasing intolerable communication cost.

2) Our proposed protocol for the PPTAM problem has lower computation and communication costs than the related solution derived by the techniques in [62], while keeping the same level of SMC security.

The remainder of the chapter is organized as follows: Section 4.2 discusses some related work. Section 4.3 lists the necessary preliminaries for our protocols. Section 4.4 and Section 4.5 propose the PPTM and PPTAM protocols respectively, and prove their correctness. In Section 4.6 we analyze the security for both protocols. In Section 4.7 we compare our protocols with the related work considering the computation and communication costs, and get their response times by experiments in a moderate-scale application. Section 4.8 concludes the whole chapter and discuss some open problems.

4.2 Related Work

Both PPTM and PPTAM are specific problems belonging to the general *Secure Multiparty Computation* (SMC) problem. There have been general solutions for the SMC problem ([49], [92]). In general SMC, the function to be computed is represented by a circuit, and every gate of the circuit is privately evaluated. However, when this general solution is used for a specific problem, the large size of the circuit and high cost of evaluating all gates will result in a much less efficient protocol than the non-private protocol for this problem. For specific problems, their properties can be utilized to produce more efficient private protocols.

PPTM and PPTAM can be traced back to the specific problem of private equality test (PET) in two-party case, where each party has a single element and wants to test whether they are equal without publishing the elements. The problem of PET was considered in [9], [29], [67] and [74]. Another related problem is for the situation where each party holds a dataset, and privately computes the intersection (PI) of two datasets ([74] and [36]). The elements other than those in the intersection should be kept private. By these two-party case solutions, one party will know the specific matching times of its elements on the other party, so they are unsuitable to be used for our N-party case PPTM and PPTAM, otherwise too much sensitive information will be leaked, e.g., the second privacy requirement in Section 4.1 will be breached.

Privacy preservation issues have also been considered for approximation of a function f among tuples owned by multiple parties. The function f may be Euclidean distance ([21],[30],[56]), set difference ([36]), Hamming distance ([21],[56]) and scalar product (reviewed in [46]). All of these problems assume that for some integer U, Alice has $q \in [U]^d$,

Bob has $P = p_1, ..., p_n \in [U]^d$. Alice should learn those p_i such that $f(q, p_i) \leq m$, where m is a threshold on the distance. These problems have different privacy requirements from ours. In these problems Alice will know how many p_i on Bob satisfy $f(q, p_i) \leq m$, but for our PPTM and PPTAM the specific number of p_i should be kept private, and we only care about the fact whether there are some p_i. Therefore, the solutions for these problems are not suitable for our PPTM and PPTAM, otherwise the specific matching times and matching orignations will be leaked.

4.2.1 Related Work for PPTM

The PPTM problem can be considered as computing a function $\bigcup_{i'=1,...,N,i\neq i'}(T_i \cap T_{i'})$ on P_i for $i = 1, ..., N$. In [62] privacy preserving set intersection and set union on N ($N \geq 2$) parties are solved using the encrypted polynomial techniques. Thus we can derive a solution from their work: on each P_i, 1) $T_i \cap T_{i'}$ for $i' = 1, ..., N$ and $i \neq i'$ are computed by their set intersection protocol. 2) without disclosing the results of these intersections, $\bigcup_{i'=1,...,N,i\neq i'}(T_i \cap T_{i'})$ is computed by their set union protocol. We name this solution *Solution D1* in this chapter.

In [63] the problem of *Threshold Contribution Threshold Set-union* (TCTS) was solved, where each party learns those elements which appear both in his private input and the threshold set. The threshold set is composed by those elements which appear at least t times in the union of all parties' sets. If there are no duplicate tuples inside any party's own database, a protocol for TCTS can be used to solve the PPTM problem by setting $t = 2$.

Both Solution D1 and TCTS protocol are correct with *overwhelming probability* when determining whether an element $T(i, j) \in \bigcup_{i'=1,...,N,i\neq i'}(T_i \cap T_{i'})$. An overwhelming probability is near one, and typically it is a probability $\geq 1 - 2^{-80}$. In the two solutions wrong determinations happen with *negligible probability*. A negligible probability is near zero and typically $\leq 2^{-80}$. The reason for the wrong determinations is that a randomly selected polynomial in these solutions may have a root which is rightly in the tuple set of the parties. We will prove that our protocol for PPTM is also overwhelmingly correct (in Lemma 8 and 9). We will compare our PPTM protocol with the two related solutions by their computation and communication costs in Section 4.7.

4.2.2 Related Work for PPTAM

We use TN_k^i for $k = 1, ..., M$ to denote the tuple set $\{T(i, j)|T(i, j)_k \textit{ appears at least } 2 \textit{ times in } A_k\}$ on P_i, then the PPTAM problem is for P_i to compute $TN_1^i \cap TN_2^i \cap ... \cap TN_M^i$ $(= T1_i)$.
In [62] the threshold set-union protocol is to find out the elements which appear in the union of the players' private input multisets at least a threshold number t times. Thus, a derived solution from the techniques in [62] can be: 1) on each P_i, TN_k^i is computed for the k-th attribute using their threshold set union protocol for $k = 1, ..., M$, by setting $t = 2$; 2) without disclosing the results of the set unions, $TN_1^i \cap TN_2^i \cap ... \cap TN_M^i$ is computed by their set intersection protocol. We name this solution *Solution D2* in this chapter. Solution D2 is correct with overwhelming probability according to [62]. We will prove our protocol for PPTAM is also overwhelmingly correct (in Lemma 10 and 11), and compare the two solutions by their computation and communication costs in Section 4.7.

4.3 Preliminaries

4.3.1 Adversary Model

There are two types of adversaries in Secure Multi-party Computation, as described in Chapter 1. In this chapter we assume the parties are semi-honest, and they may compose any coalition of c ($c \leq N-1$) parties $(P_{i_1}, ..., P_{i_c})$. In Definition 2 of Chapter 1, we have set the standard for what it meand for a multi-party protocol to *privately compute* a function f. We will prove the security of our protocols for PPTM and PPTAM according to this definition.

4.3.2 Homomorphic Encryption

Our protocols are based on an additive Homomorphic Encryption (HE) scheme. Let ε be a probabilistic encryption scheme. Let M be the message space and C the ciphertext space such that M is a group under operation $\oplus$ and C is a group under operation $\odot$. ε is a $(\oplus, \odot)$-HE scheme if for any instance $E_R(\cdot)$ of the encryption scheme, given $c_1 = E_{r_1}(m_1)$ and $c_2 = E_{r_2}(m_2)$, there exists an r such that $c_1 \odot c_2 = E_{r_1}(m_1) \odot E_{r_2}(m_2) = E_r(m_1 \oplus m_2)$. ε is *additive* when it is a $(+, \odot)$ scheme, and *multiplicative* when it is a $(*, \odot)$ scheme.

The HE scheme in our protocols is also required to support secure (N, N)-threshold decryption. The corresponding secret key is shared by a group of N parties, and the decryption is unable to be performed by any single party, unless all parties act together.

Thus, we use Paillier's cryptosystem ([75]) for its following properties: 1) it is an additive homomorphic encryption scheme. Given two encryptions $E(m_1)$ and $E(m_2)$, $E(m_1 + m_2) = E(m_1) \cdot E(m_2)$; 2) given an encryption $E(m)$ and a scalar a, $E(a \cdot m) = E(m)^a$; 3) (N, N)-threshold decryption can be supported (by [33],[34]). In this chapter, $\mathcal{N}$ is the RSA-modulus which is the multiplication of two large prime numbers, and $\mathbb{Z}_{\mathcal{N}}$ is the plaintext space of Paillier's cryptosystem.

4.3.3 Calculations on encrypted polynomials

In some blocks of our protocols, we need do some calculations on encrypted polynomials. For a polynomial $f(x) = \sum_{i=0}^{m} a_i x^i$, we use $E(f(x))$ to denote the list of encrypted coefficients $\{E(a_i) | i = 0, ..., m\}$. Given $E(f(x))$, where $E(\cdot)$ is an additive homomorphic encryption scheme (e.g., Paillier's cryptosystem), some computations can be made as follows (some of them have also been used in [62]):

1) At a value v, we can evaluate $E(f(x))$: $E(f(v)) = E(a_m v^m + a_{m-1} v^{m-1} + ... + a_0) = E(a_m)^{v^m} E(a_{m-1})^{v^{m-1}} \cdots E(a_0)$.

2) Given $E(f(x))$, we can compute $E(c \cdot f(x)) = \{E(a_m)^c, ..., E(a_0)^c\}$.

3) Given $E(f(x))$ and $E(g(x))$, $g(x) = \sum_{j=0}^{m} b_j x^j$, we can compute $E(f(x) + g(x)) = \{E(a_m)E(b_m), ..., E(a_0)E(b_0)\}$.

4) Given $f(x)$ and $E(g(x))$, we can compute $E(f(x) * g(x))$. Suppose that $g(x) = \sum_{j=0}^{n} b_j x^j$, $f(x) * g(x) = \sum_{k=0}^{m+n} c_k x^k$, then $E(c_k) = E(a_0 b_k + a_1 b_{k-1} + ... + a_k b_0) = E(b_k)^{a_0} \cdots E(b_0)^{a_k}$. a_i or b_j are treated as zero if $i > m$ or $j > n$.

Table 4.2: Major Notations in This chapter

Notation	*Definition*
N	Total number of parties
S	Total number of tuples on each party
M	Total number of attributes of each tuple
P_i	The i-th party
c	Total number of colluded parties
I	The index set of colluded parties,$\{i_1, ..., i_c\}$
I'	The index set of honest parties, $\{1, ..., N\} \setminus I$
$T(i,j)$	The j-th tuple on P_i
$T(i,j)_k$	The k-th attribute value of $T(i,j)$
T_i	The tuple set on P_i, $\{T(i,j)\|j = 1, ..., S\}$
A_k^i	The value set of the k-th attribute on P_i, i.e., $\{T(i,j)_k\|j = 1, ..., S\}$
A_k	The value set of the k-th attribute on the N parties, i.e., $\bigcup_{i=1}^{N} A_k^i = \{T(i,j)_k\|i = 1, ..., N,\ j = 1, ..., S\}$
A_k^I	The value set of the k-th attribute on the colluded parties, i.e., $\bigcup_{i\in I} A_k^i = \{T(i,j)_k\|i \in I, j = 1, ..., S\}$
$A_k^{I'}$	The value set of the k-th attribute on the honest parties, i.e., $\bigcup_{i\in I'} A_k^i = \{T(i,j)_k\|i \in I', j = 1, ..., S\}$
$T1_i$	The tuple set $\{T(i,j)\}$ in which $\forall k \in \{1, ..., M\}$, $T(i,j)_k$ of $T(i,j)$ appear at least 2 times in A_k
$T0_i$	The tuple set $\{T(i,j)\}$ in which $\exists M' \subseteq \{1, ..., M\}$, $\forall k \in M'$, $T(i,j)_k$ appear only one time in A_k
f_i	The polynomial constructed on T_i, used in Section 4.4. $f_i = (x - T(i,1)) \cdots (x - T(i,S))$, $i \in \{1, ..., N\}$.
f_k	The polynomial constructed on A_k, used in Section 4.5. $f_k = \prod_{i=1}^{N} \prod_{j=1}^{S} (x_k - T(i,j)_k)$, $k \in \{1, ..., M\}$.
f_k^i	The polynomial constructed on A_k^i. $f_k^i = \prod_{j=1}^{S} (x_k - T(i,j)_k)$
$f_k^{(1)}$	The first derivative of f_k
F_k	The polynomial constructed on A_k^I, used in Section 4.5. $F_k = \prod_{i\in I} f_k^i$
G_k	The polynomial constructed on $A_k^{I'}$, used in Section 4.5. $G_k = \prod_{i\in I'} f_k^i$
$\mathbb{Z}_{\mathcal{N}}$	The plaintext space of Paillier's cryptosystem

5) The first derivative of $f(x)$ is $f^{(1)}(x) = \sum_{i=0}^{m-1}(i+1)a_{i+1}x^i$. Given $E(f(x))$, we can get $E(f^{(1)}(x)) = \{E(c_i)|i = 0, ..., m-1\}$: $E(c_i) = E((i+1)a_{i+1}) = E(a_{i+1})^{(i+1)}$.

6) Suppose that $f_k(x_k) = \sum_{i=0}^{m} a_{k,i}x_k^i$ for $k = 1, ..., n$, and $G(x_1, x_2, ..., x_n) = f_1(x_1) + f_2(x_2) + ... + f_n(x_n) = \sum_{k=1}^{n}\sum_{i=0}^{m} a_{k,i}x_k^i$. Given $E(f_k(x_k))$ for $k = 1, ..., n$, we can get $E(G(x_1, x_2, ..., x_n)) = \{E(a_{k,i})|k = 1, ..., n, i = 0, ..., m\} = \{E(f_1(x_1)), E(f_2(x_2)), ..., E(f_n(x_n))\}$.

7) At a point $(v_1, v_2, ..., v_n)$, we can evaluate $E(G(x_1, x_2, ..., x_n))$ which is computed in 6): $E(G(v_1, v_2, ..., v_n)) = \prod_{k=1}^{n} E(f_k(v_k))$.

4.3.4 Notations

If not explained otherwise, the major notations in this chapter have the definitions in Table 4.2.

4.4 Privacy Preserving Tuple Matching

4.4.1 Main Idea

Denoting $T(i,j)$ as $T(i,j)_1||...||T(i,j)_M$ ("||" is concatenation of bitstrings), P_i can compute a polynomial f_i to represent its inputs (tuple set T_i): $f_i = (x - T(i,1)) \cdots (x - T(i,S))\ mod\ \mathcal{N}$. Then each coefficient of f_i is in the Paillier plaintext ring $\mathbb{Z}_{\mathcal{N}}$, and is encrypted to get $E(f_i)$. If $T(i,j)$ has a duplicate on some $P_{i'}$ ($i' \neq i$), its evaluation on the polynomial $f_i * r + f_{i'} * r'$ is 0, otherwise the evaluation is a random number, given r and r' are polynomials with random coefficients. Then the N parties can collaboratively

compute the following polynomial for each P_i:

$$F_i = \prod_{i'=1...N, i' \neq i} (f_i * \sum_{k=1}^{N} r_{i'k} + f_{i'} * \sum_{k=1}^{N} r'_{i'k}) \tag{4.1}$$

$r_{i'k}$ and $r'_{i'k}$ are generated by P_k: $r_{i'k} = \sum_{j=0}^{\alpha} a_j x^j$, $r'_{i'k} = \sum_{j=0}^{\alpha} a'_j x^j$. a_j and a'_j for $j = 0, ..., \alpha$ are uniformly selected from $\mathbb{Z}_{\mathcal{N}}$. The coefficients of F_i are encrypted in the computation for security, and only its evaluations on $T(i, j)$ are decrypted. By Lemma 8 and 9, we can use the evaluation $F_i(T(i, j))$ to determine whether $T(i, j)$ has any duplicate with the other parties.

The degrees of $r_{i'k}$ and $r'_{i'k}$ should be high enough to resist coefficient analysis, but low enough to reduce computation time. We give a simple discussion below that the minimum and secure value of α is $\lceil \frac{S}{N-1} \rceil$, and its suitability will be shown in Lemma 12 in detail.

Suppose there are c ($c \leq N - 1$) colluded parties including P_i, whose index set is denoted by I. P_i gets S evaluations $F_i(T(i, j))$ and may analysis the coefficients of F_i by interpolation. From P_i's view, $f_i(T(i, j)) = 0$, then the evaluations $F_i(T(i, j)) = f_{\mathcal{I}} f_{\mathcal{I}'} \prod_{i'=1...N, i' \neq i} \sum_{k=1}^{N} r'_{i'k}$, in which $f_{\mathcal{I}}$ and $f_{\mathcal{I}'}$ represent the inputs of the $c - 1$ parties colluded with P_i, and inputs of the honest parties for $\mathcal{I}' = \{1, ..., M\} \setminus I$, respectively. The degrees of $f_{\mathcal{I}'}$ and $\prod_{i'} \sum_{k=1}^{N} r'_{i'k}$ are $(N - c)S$ and $(N - 1)\alpha$, and the leading coefficient of $f_{\mathcal{I}'}$ is 1, so P_i has $(N-c)S+(N-1)\alpha+1$ unknown coefficients in each $F_i(T(i, j))$. Besides the S evaluations, P_i may possibly get evaluations from its coalition: for any $i'' \in I$, if $F_{i''}(T(i'', j)) = 0$ and the colluded parties except $P_{i''}$ have no duplicate of $T(i'', j)$, then it is must be $P_{\mathcal{I}'}$ that have a duplicate, and thus $f_{\mathcal{I}'}(T(i'', j)) = 0$. The worst case is that the coalition knows $(N - c)S$ numbers of $T(i'', j)$ of such kind, i.e., knows all inputs of $P_{\mathcal{I}'}$. In this case, P_i knows $S + (N - c)S$ evaluations of F_i. To resist coefficient analysis on F_i, there should be $S + (N - c)S < (N - c)S + (N - 1)\alpha + 1$, thus $\alpha \geq \lceil \frac{S}{N-1} \rceil$.

Lemma 8 *If $T(i, j)$ has a duplicate with any $P_{i'}$ ($i' \in \{1, ..., N\} \setminus \{i\}$), then the evaluation $F_i(T(i, j)) = 0$.*

Proof: If some $P_{i'}$ has a duplicate of $T(i, j)$, then both $f_{i'}(T(i, j))$ and $f_i(T(i, j))$ are 0, and $F_i(T(i, j)) = 0$. ■

Lemma 9 *If the evaluation $F_i(T(i, j)) = 0$, overwhelmingly $T(i, j)$ has a duplicate with some $P_{i'}$ ($i' \in \{1, ..., N\} \setminus \{i\}$).*

Proof: If $F_i(T(i, j)) = 0$, there must be at least one factor $f_i(T(i, j)) * \sum_{k=1}^{N} r_{i'k} + f_{i'}(T(i, j)) * \sum_{k=1}^{N} r'_{i'k} = 0$. In this factor, $f_i(T(i, j))$ must be 0. To prove $f_{i'}(T(i, j)) = 0$ we need prove that overwhelmingly $\sum_{k=1}^{N} r'_{i'k} \neq 0$. Suppose $\sum_{k=1}^{N} r'_{i'k} = \sum_{j=0}^{\alpha} b_j x^j$, then b_j for $j = 0, ..., \alpha$ are uniformly distributed over $\mathbb{Z}_{\mathcal{N}}$, $\sum_{k=1}^{N} r'_{i'k}(T(i, j))$ is also uniformly distributed over $\mathbb{Z}_{\mathcal{N}}$. That is, $\sum_{k=1}^{N} r'_{i'k}(T(i, j)) = 0$ with probability $1/\mathcal{N}$. When $\mathcal{N}$ is large enough (e.g., 1024 bits), overwhelmingly (e.g., with a probability of $1 - 2^{-1024}$) $T(i, j)$ is not a root of $\sum_{k=1}^{N} r'_{i'k}$, and then is a root of $f_{i'}$, and the corresponding $P_{i'}$ has a duplicate of $T(i, j)$. ■

4.4.2 The Protocol

Protocol 1: *Protocol for Privacy Preserving Tuple Matching*

Inputs: A database with M ($M \geq 2$) attributes is horizontally partitioned among N ($N \geq 2$) parties. Every party has S private tuples. Every party holds the public key and it is own share of the secret key for the threshold Paillier's cryptosystem.

Output: Each party P_i knows whether its tuples belong to $T_i \cap (\bigcup_{i'=1...N, i' \neq i} T_{i'})$.

Steps :

1) Each party P_i computes its $f_i = (x - T(i,1)) \cdots (x - T(i,S)) \ mod \ \mathcal{N}$.
2) P_1 initializes $E(F_{1,1}) = E(1)$, and repeats the following for $j = 2, ..., N$.
   2.1) P_1 computes $E(F_{1,(j-1)} * f_1)$ and sends it to all the other parties. Each party P_k ($k \neq 1$) randomly chooses r_{jk} as described in Section 4.4.1, computes $E(F_{1,(j-1)} * f_1 * r_{jk})$ by computation 4) in Section 4.3.3, and sends it back to P_1. P_1 also randomly chooses r_{j1} and computes $E(F_{1,(j-1)} * f_1 * \sum_{k=1}^{N} r_{jk})$ by computation 3) in Section 4.3.3.
   2.2) P_1 sends $E(F_{1,(j-1)})$ to P_j, P_j computes $E(F_{1,(j-1)} * f_j)$, sends it to all the other parties. Each of these parties P_k, including P_j, randomly chooses r'_{jk}, computes $E(F_{1,(j-1)} * f_j * r'_{jk})$ and sends it to P_1. P_1 computes $E(F_{1,(j-1)} * f_j * \sum_{k=1}^{N} r'_{jk})$.
   2.3) P_1 computes $E(F_{1,j}) = E(F_{1,(j-1)}(f_1 * \sum_{k=1}^{N} r_{jk} + f_j * \sum_{k=1}^{N} r'_{jk}))$ by summing $E(F_{1,(j-1)} * f_1 * \sum_{k=1}^{N} r_{jk})$ and $E(F_{1,(j-1)} * f_j * \sum_{k=1}^{N} r'_{jk})$.

   At the end of $j = N$, P_1 gets $E(F_1) = E(F_{1,N}) = E(\prod_{j=2}^{N}(f_1 * \sum_{k=1}^{N} r_{jk} + f_j * \sum_{k=1}^{N} r'_{jk}))$.
3) Each P_i other than P_1 repeats Step 2) and gets $E(F_i) = E(\prod_{i'=1...N, i' \neq i}(f_i * \sum_{k=1}^{N} r_{i'k} + f_{i'} * \sum_{k=1}^{N} r'_{i'k}))$.
4) Each P_i evaluates $E(F_i)$ at $T(i,j)$ for $j = 1, ..., S$, using computation 1) in Section 4.3.3.
5) Each party decrypts $E(F_i(T(i,j)))$ in the collaboration of the other $N-1$ parties for $j = 1, ..., S$. If $F_i(T(i,j)) = 0$, $T(i,j)$ has a duplicate with the other parties; otherwise, $T(i,j)$ has no duplicate with the other parties.

In the above protocol, each party P_i computes its $E(F_i)$ in $N-1$ rounds. For example, P_1 firstly computes $E(F_{1,2}) = E(f_1 * \sum_{k=1}^{N} r_{2k} + f_2 * \sum_{k=1}^{N} r'_{2k})$ in Step 2) with $j = 2$, by summing $E(f_1 * \sum_{k=1}^{N} r_{2k})$ and $(f_2 * \sum_{k=1}^{N} r'_{2k})$. Then P_1 repeats Step 2) with $j = 3$, computes $E(F_{1,3}) = E(F_{1,2}(f_1 * \sum_{k=1}^{N} r_{3k} + f_3 * \sum_{k=1}^{N} r'_{3k}))$, by summing $E(F_{1,2} f_1 * \sum_{k=1}^{N} r_{3k})$ and $E(F_{1,2} f_3 * \sum_{k=1}^{N} r'_{3k})$. When $j = N$, P_1 gets $E(F_1) = E((f_1 * \sum_{k=1}^{N} r_{2k} + f_2 * \sum_{k=1}^{N} r'_{2k}) \cdot \cdot \cdot (f_1 * \sum_{k=1}^{N} r_{Nk} + f_N * \sum_{k=1}^{N} r'_{Nk}))$. P_1 evaluates each $E(F_1(T(1,j)))$, and decrypts it to see whether it is 0.

The correctness of Protocol 1 is obvious by Lemma 8 and Lemma 9. From Lemma 9, Protocol 1 may have wrong determinations if $T(i,j)$ is a root of the random polynomial $\sum_{k=1}^{N} r'_{i'k}$, but this happens with negligible probability (e.g., 2^{-1024}, if $|\mathcal{N}| = 1024$).

4.5 Privacy Preserving Threshold Attributes Matching

4.5.1 Main Idea

As defined in Section 4.1, for the PPTAM problem, each P_i needs to privately determine whether $T(i,j) \in T1_i$. We will treat each attribute of a tuple as an individual input. The value set of the k-th attribute on P_i is $A_k^i = \{T(i,j)_k | j = 1, ..., S\}$, and the value set of the k-th attribute on N parties is $A_k = \bigcup_{i=1}^{N} A_k^i$. We can construct the polynomial $f_k^i(x_k) = \prod_{j=1}^{S}(x_k - T(i,j)_k)\ mod\ \mathcal{N}$ for A_k^i, and $f_k(x_k) = \prod_{i=1}^{N} f_k^i(x_k)\ mod\ \mathcal{N}$ for A_k. $f_k^{(1)}(x_k)$ is the first derivative of $f_k(x_k)$.

Our PPTAM protocol is based on the following "iff" relation: an element a appears in the multiset $\{a_1, ..., a_n\}$ at least 2 times $\Longleftrightarrow$ a is a root of polynomial g_n and $g_n^{(1)}$, in which $g_n = (x - a_1) \cdots (x - a_n)$, $g_n^{(1)}$ is the first derivative of g_n. "$\Longrightarrow$" is easy to verify. "$\Longleftarrow$" can be verified by mathematical induction: 1) when $n = 2$, if $g_2^{(1)}(a_1) = 0$, then $a_1 = a_2$; 2) when $n > 2$, $g_n^{(1)} = g_{n-1}^{(1)} \cdot (x - a_n) + g_{n-1}$. If a_n is a root of $g_n^{(1)}$, then $g_{n-1}(a_n) = 0$, i.e., some a_i $(1 \le i \le n-1) = a_n$. If some a_i $(1 \le i \le n-1)$ is a root of $g_n^{(1)}$, then a_i is a root of $g_{n-1}^{(1)}$ or $a_i = a_n$. In both cases, a_i appears at least 2 times in the multiset.

However, it is unsuitable to simply determine whether $T(i,j)_k$ appears at least 2 times in each attribute A_k, and combine the M results to know whether $T(i,j) \in T1_i$, otherwise P_i will know the matching states on each attribute of $T(i,j)$ if $T(i,j)$ is not in $T1_i$ (breaching the third privacy requirement in Section 4.1).

In protocol 2 of Section 4.5.2, to determine whether $T(i,j) \in T1_i$, we firstly compute $f_k^{(1)}$ for $k = 1, ..., M$, and randomize them as $f_k^{(1)} * \sum_{i=1}^{N} s_{ik}$ for $k = 1, ..., M$, in which $s_{ik} = \sum_{j=0}^{\alpha} a_j x_k^j$, a_j for $j = 0, ..., \alpha$ are randomly chosen from $\mathbb{Z}_{\mathcal{N}}$ by P_i. $Degree(s_{ik})$ should be high enough so that $f_k^{(1)} * \sum_{i=1}^{N} s_{ik}$ resists root analysis, but low enough to reduce the computation time. we make $\alpha = (N-1)S$ and will show its suitability in Lemma 13.

Secondly we construct a vector of multivariate polynomials $\mathcal{G}(x_1, ..., x_M) = (g_1(x_1, ..., x_M), ..., g_M(x_1, ..., x_M)) = \mathcal{F} \cdot \mathcal{M}$, in which $\mathcal{F} = (f_1^{(1)} * \sum_{i=1}^{N} s_{i1}, ..., f_M^{(1)} * \sum_{i=1}^{N} s_{iM})$, $\mathcal{M}$ is an $M \times M$ nonsingular matrix. By Lemma 10 and 11 we can use the evaluation of $T(i,j)$ at $\mathcal{G}(x_1, ..., x_M)$ to determine whether $T(i,j) \in T1_i$.

How to ensure the nonsingularity of M in the computation is a critical problem. In our Protocol 2 we make each party P_i compute the product $\mathcal{R}_i\mathcal{U}$, where $\mathcal{R}_i$ is an upper triangular matrix which is nonsingular and generated by P_i, $\mathcal{U}$ is a common nonsingular matrix. Then they compute $E(\mathcal{F} \cdot \mathcal{R}_i\mathcal{U})$, add them, and get $E(\mathcal{F} \cdot (\sum_{i=1}^{N} \mathcal{R}_i)\mathcal{U})$. Let $\mathcal{M} = (\sum_{i=1}^{N} \mathcal{R}_i)\mathcal{U}$, because $\sum_{i=1}^{N} \mathcal{R}_i$ is nonsingular, $\mathcal{M}$ is also nonsingular.

4.5.2 The Protocol

Protocol 2: *Protocol for Privacy Preserving Threshold Attributes Matching*

Inputs: A database with M $(M \ge 2)$ attributes is horizontally partitioned among N $(N \ge 2)$ parties. Every party has S private tuples. Every party holds the public key and its own share of the secret key for the threshold Paillier's cryptosystem.

Output: Each party P_i knows which each tuple of its own belongs to $T1_i$ or $T0_i$.

1) Computing $E(\mathcal{F})$:

1.1) For every attribute $k = \{1, ..., M\}$, P_1 computes $F_k^1(x_k) = f_k^1(x_k) = \prod_{j=1}^{S}(x_k - T(1,j)_k)\ mod\ \mathcal{N}$ and $E(F_k^1(x_k))$. P_1 sends $E(F_k^1(x_k))$ to P_2.

1.2) For $i = 2, ..., N$,

1.2.1) For $k = \{1, ..., M\}$, P_i computes $f_k^i(x_k) = \prod_{j=1}^{S}(x_k - T(i,j)_k)\ mod\ \mathcal{N}$ and then, with $E(F_k^{i-1}(x_k))$ from P_{i-1}, computes $E(F_k^i(x_k)) = E(F_k^{i-1}(x_k) * f_k^i(x_k))$, following computation 4) of Section 4.3.3.

1.2.2) For $k = \{1, ..., M\}$, P_i sends $E(F_k^i(x_k))$ to $P_{(i+1\ mod\ N)}$.

1.3) P_1 gets $E(f_k(x_k)) = E(\prod_{i=1}^{N}\prod_{j=1}^{S}(x_k - T(i,j)_k))$ from P_N for $k = \{1, ..., M\}$. P_1 computes the first derivative of $E(f_k)$, i.e., $E(f_k^{(1)})$, following computation 5) of Section 4.3.3.

1.4) P_1 sends all $E(f_k^{(1)})$ to $P_2, P_3, ..., P_N$, for $k = \{1, ..., M\}$.

1.5) Every P_i generates $s_{ik} = \sum_{j=0}^{\alpha} a_j x_k^j$ by the method in Section 4.5.1, and computes $E(f_k^{(1)} * s_{ik})$ for $k = 1, ..., M$ by computation 4) of Section 4.3.3, and sends them to the other parties.

1.6) Every P_i computes $E(f_k^{(1)} * \sum_{i=1}^{N} s_{ik})$ for $k = 1, ..., M$, by computation 3) of Section 4.3.

2) Computing $E(\mathcal{G})$:

2.1) Party P_1 generates a nonsingular $M \times M$ matrix $\mathcal{U}$ over $\mathbb{Z}_N$ as follows (by the method in [79]), and sends it to the other parties.

$$\begin{pmatrix} u_{11} & u_{12} & ... & u_{1M} \\ u_{21} & u_{22} & ... & u_{2M} \\ \cdots & & & \\ u_{M1} & u_{M2} & ... & u_{MM} \end{pmatrix} \tag{4.2}$$

2.2) Every party P_i generates an upper triangular matrix $\mathcal{R}_i$ as follows:

$$\begin{pmatrix} r_{11}^i & r_{12}^i & ... & r_{1M}^i \\ 0 & r_{22}^i & ... & r_{2M}^i \\ \cdots & & & \\ 0 & 0 & ... & r_{MM}^i \end{pmatrix} \tag{4.3}$$

in which every entry in the upper triangularity $r_{kl}^i (k \leq l)$ is uniformly selected from $\mathbb{Z}_N$, and the diagonal entries $r_{kl}^i (k = l)$ are nonzero.

2.3) With $E(\mathcal{F}) = (\ E(f_1^{(1)} * \sum_{i=1}^{N} s_{i1}), ..., E(f_M^{(1)} * \sum_{i=1}^{N} s_{iM})\ )$, every party P_i computes $E(\mathcal{F}\mathcal{R}_i\mathcal{U})$ according to computation 2) and 6) in Section 4.3.

2.4) Every party P_i exchanges their $E(\mathcal{F}\mathcal{R}_i\mathcal{U})$ with the other parties, then computes $E(\mathcal{G}) = E(\sum_{i=1}^{N} \mathcal{F}\mathcal{R}_i\mathcal{U}) = E(\mathcal{F}(\sum_{i=1}^{N} \mathcal{R}_i)\mathcal{U})$ according to computation 3) in Section 4.3.

$E(\mathcal{G})$ is a vector of encrypted multivariate polynomials ($E(g_1), ..., E(g_M)$). In the following equation, $R_{v,w}$ $(v = 1, ..., M,\ w = 1, ..., M)$ is the (v, w)-th entry of $(\sum_{i=1}^{N} \mathcal{R}_i)\mathcal{U}$.

$$
\begin{aligned}
E(\mathcal{G}) &= (\ E(g_1), ..., E(g_M)\ ) \\
&= \Bigg(\ E\big(R_{11}f_1^{(1)}\sum_{i=1}^{N} s_{i1} + R_{12}f_2^{(1)}\sum_{i=1}^{N} s_{i2} + ... + R_{1M}f_M^{(1)}\sum_{i=1}^{N} s_{iM}\big), \\
&\quad ..., \\
&\quad E\big(R_{M1}f_1^{(1)}\sum_{i=1}^{N} s_{i1} + R_{M2}f_2^{(1)}\sum_{i=1}^{N} s_{i2} + ... + R_{MM}f_M^{(1)}\sum_{i=1}^{N} s_{iM}\big) \Bigg)
\end{aligned}
\tag{4.4}
$$

3) Evaluation and Decryption :

3.1) On every P_i, for $j = \{1, ..., S\}$, $E(\mathcal{G})$ is evaluated at the tuple $T(i, j) = (\ T(i, j)_1, ..., T(i, j)_M\ )$ to get $Y(i, j) = E(g_1(T(i, j)_1, ..., T(i, j)_M), ..., g_M(T(i, j)_1, ..., T(i, j)_M))$, following computation 7) of Section 4.3.3.

3.2) Every P_i decrypts $Y(i, j)$ for $j = \{1, ..., S\}$ with the help of the other parties. If $D(Y(i, j)) = (0, ..., 0)$, the corresponding $T(i, j) \in\ T1_i$; otherwise, $T(i, j) \in\ T0_i$.

We prove the correctness of Protocol 2 in the following two lemmas:

Lemma 10 *Given a tuple $T(i, j)$, if $T(i, j)_k$ appears at least 2 times in A_k for $k = 1, ..., M$, then the evaluation of $g_l(T(i, j)_1, ..., T(i, j)_M)$ equals 0 for $l = 1, ..., M$.*

Proof: If $T(i, j)_k$ appears at least 2 times in A_k for $k = 1, ..., M$, $T(i, j)_k$ is a root of $f_k^{(1)}(x_k)$ for $k = 1, ..., M$. Then $\mathcal{F}(T(i, j)) = (f_1^{(1)}(T(i, j)_1) \sum_{i=1}^{N} s_{i1}, ..., f_M^{(1)}(T(i, j)_M) \sum_{i=1}^{N} s_{iM}) = (0, ..., 0)$, $\mathcal{G} = \mathcal{F}(\sum_{i=1}^{N} \mathcal{R}_i)\mathcal{U} = (0, ..., 0)$. ■

Lemma 11 *Given a tuple $T(i, j)$, if the evaluation of $g_l(T(i, j)_1, ..., T(i, j)_M)$ equals 0 for $l = 1, ..., M$, overwhelmingly $T(i, j)_k$ appears at least 2 times in A_k for $k = 1, ..., M$.*

Proof: If $\mathcal{G}(T(i, j)) = (0, ..., 0)$, to prove $\mathcal{F}(T(i, j)) = (0, ..., 0)$ we need firstly prove $(\sum_{i=1}^{N} \mathcal{R}_i)\mathcal{U}$ is nonsingular. In matrix $\mathcal{R}_i$ $(i = 1, ..., N)$, $r_{kl}^i(k = l) > 0$, so $rank(\sum_{i=1}^{N} \mathcal{R}_i) = rank(\mathcal{R}_i) = M$. $\mathcal{U}$ is also generated to be full rank. Therefore $rank\big(\ (\sum_{i=1}^{N} \mathcal{R}_i)\mathcal{U}\ \big) = M$. If $\mathcal{G}(T(i, j)) = (0, ..., 0)$, a linear system $\mathcal{F}(\sum_{i=1}^{N} \mathcal{R}_i)\mathcal{U} = (0, ..., 0)$ can be made, and it has only one solution: $\mathcal{F}(T(i, j)) = (0, ..., 0)$, i.e.,$f_i^{(1)}(T(i, j)_k) * \sum_{i=1}^{N} s_{ik}(T(i, j)_k) = 0$ for $k = 1, ..., M$. Similarly with the proof of Lemma 9, $\sum_{i=1}^{N} s_{ik}$ is a polynomial uniformly distributed over the polynomial ring $\mathbb{Z}_{\mathcal{N}}[x]$, and with probability $\frac{1}{\mathcal{N}}$, $T(i, j)_k$ is a root of it. When $\mathcal{N}$ is large enough (e.g., 1024 bits), overwhelmingly (e.g., with a probability of $1 - 2^{-1024}$) it is not $\sum_{i=1}^{N} s_{ik}(T(i, j)_k) = 0$, but $f_i^{(1)}(T(i, j)_k) = 0$. Thus

if $\mathcal{G}(T(i,j)) = (0,...,0)$, overwhelmingly $T(i,j)_k$ is a root of $f_k^{(1)}$ for $k = 1,...,M$, and $T(i,j)_k$ appears at least 2 times in A_k for $k = 1,...,M$. ■

From Lemma 11, Protocol 2 may have wrong determinations if $T(i,j)_k$ is a root of the random polynomial $\sum_{i=1}^{N} s_{ik}$, but this happens with negligible probability (e.g., 2^{-1024}, if $|\mathcal{N}| = 1024$).

4.6 Security Analysis

4.6.1 Security Analysis for Protocol 1

The Inferred Information by the Definition of PPTM

If there is any coalition of c ($c \leq N-1$) semi-honest parties P_I ($\{P_i | i \in I = \{i_1,...,i_c\}\}$), by the definition of PPTM, it is unavoidable for P_i ($i \in I$) to infer the following information by combining inputs and outputs of its coalition parties:

1) if the determination is $T(i,j)$ has a duplicate with the other parties, and P_i knows $T(i,j)$ also has a duplicate with P_I, then P_i is incapable of knowing whether there is any duplicate of $T(i,j)$ with the remaining parties $P_{I'}$ ($I' = \{1,...,N\} \setminus I$).

2) if the determination is $T(i,j)$ has a duplicate with the other parties, and P_i knows $T(i,j)$ has no duplicate with P_I, then it knows that $T(i,j)$ must have a duplicate with $P_{I'}$; We denote these $T(i,j)$ on $P_{I'}$ as $\mathcal{T}$. The worst situation is P_i knows all tuples on $P_{I'}$, but this requires $c \geq N/2$ and $|\mathcal{T}|$ is rightly $(N-c)S$, so it seldom happens.

3) if the determination is $T(i,j)$ has no duplicate with the other parties, then P_i knows that $T(i,j)$ has no duplicate with $P_{I'}$, that is, $P_{I'}$ can not have $T(i,j)$. We denote these $T(i,j)$ as $\mathcal{T}'$.

Therefore, by the definition P_i knows $\mathcal{T}$ and $\mathcal{T}'$ as illustrated above.

The Inferred Information After Participating in Protocol 1

In Lemma 12, we prove that Protocol 1 is robust against the coefficient analysis of P_i on F_i by interpolation.

Lemma 12 *In Protocol 1, any P_i in the coalition of c ($c \leq N-1$) parties ($P_{i_1},...,P_{i_c}$) can get only the following information:*

1) $\mathcal{T}$ and $\mathcal{T}'$ which are the same tuple sets denoted in Section 4.6.1.1.

2) guessing tuples on $P_{I'}$ other than $\mathcal{T}$ and $\mathcal{T}'$, after randomly choosing at least 1 *number.*

Proof: P_i gets S pairs ($T(i,j), F_i(T(i,j))$) by evaluating F_i at $T(i,j)$. Because $f_i(T(i,j)) = 0$, then $F_i(T(i,j))$ becomes $F_i'(T(i,j))$, and $F_i' = \prod_{i'=1...N,i' \neq i}(f_{i'} * \sum_{k=1}^{N} r'_{i'k})$. If P_i can know all coefficients of F_i', it can know all roots of F_i' by polynomial factoring. Without colluding with other parties, for P_i, there are $(N-1)S$ unknown coefficients in $\prod_{i'=1...N,i' \neq i} f_{i'}$ excluding the highest order coefficient (= 1). Because $\prod_{i'=1...N,i' \neq i}(\sum_{k=1}^{N} r'_{i'k}) = \sum_{j=0}^{\beta} R_j x^j$,

$\beta = (N-1)\lceil\frac{S}{N-1}\rceil$, in this part there are at least $S+1$ unknown coefficients. Totally there are at least $(N-1)S+S+1$ unknown coefficients in F'_i. It is easy to see that $(N-1)S+S+1 > S$, so P_i is unable to find a unique F'_i that fits S pairs $(\ T(i,j), F'_i(T(i,j))\ )$.

However, P_i knows f_{i_t} for $i_t \in I$, and $r'_{i'k}$ generated by P_I. Then $F'_i = f_{\mathcal{I}} f_{\mathcal{I}'} \prod_{i'=1...N, i' \neq i} (\sum_{k \in I} r'_{i'k} + \sum_{k \in I'} r'_{i'k})$, in which $f_{\mathcal{I}} = \prod_{i_t \in I, i_t \neq i} f_{i_t}$, $f_{\mathcal{I}'} = \prod_{i'_t \in I'} f_{i'_t}$, $\sum_{k \in I} r'_{i'k}$ is generated by P_I, $\sum_{k \in I'} r'_{i'k}$ is generated by $P_{I'}$.

1) if $F'_i(T(i,j)) = 0$, and $f_{\mathcal{I}}(T(i,j)) = 0$, then P_i is unable to know any root of $f_{\mathcal{I}'}$. This accords with the case 1) in Section 4.6.1.1.

2) if $F'_i(T(i,j)) = 0$, and $f_{\mathcal{I}}(T(i,j)) \neq 0$, then P_i knows $f_{\mathcal{I}'}(T(i,j)) = 0$. All these $T(i,j)$ are the same with $\mathcal{T}$ in Section 4.6.1.1.

3) if $F'_i(T(i,j)) \neq 0$, P_i knows $f_{\mathcal{I}'}(T(i,j)) \neq 0$, i.e., $T(i,j)$ is not one root of $f_{\mathcal{I}'}$. All these $T(i,j)$ are the same with $\mathcal{T}'$ in Section 4.6.1.1.

Suppose in the case 2), all P_I know C_1 roots of $f_{\mathcal{I}'}$. $C_1 \leq (N-c)S$. The worst situation is P_I know all roots of $f_{\mathcal{I}'}$, but this needs C_1 is rightly $(N-c)S$ and seldom happens. Suppose in the case 3), P_i knows C_2 pairs $(\ T(i,j), F'_i(T(i,j))\ )$ for F'_i, then $C_2 \leq S$. Because P_i knows $f_{\mathcal{I}}(T(i,j))$, P_i can know $C_1 + C_2$ evaluations of F''_i: $F''_i = F'_i / f_{\mathcal{I}} = f_{\mathcal{I}'} \prod_{i'=1...N, i' \neq i} (\sum_{k \in I} r'_{i'k} + \sum_{k \in I'} r'_{i'k})$. For P_i, there are $(N-c)S$ unknown coefficients in $f_{\mathcal{I}'}$ excluding the highest order coefficient. In $\prod_{i'=1...N, i' \neq i} (\sum_{k \in I} r'_{i'k} + \sum_{k \in I'} r'_{i'k}) = \sum_{j=0}^{\beta} R_j x^j$, $\beta = (N-1)\lceil\frac{S}{N-1}\rceil$, there are at least $S+1$ unknown coefficients. Therefore P_i still needs to arbitrarily guess $t = (N-c)S + S + 1 - (C_1 + C_2)$ coefficients in F''_i. It is easy to see that $t \geq 1$. That is, P_i should guess at least one random number before inferring other roots in $f_{\mathcal{I}'}$ than $\mathcal{T}$ and $\mathcal{T}'$. ■

Theorem 7 *Protocol 1 is a privacy preserving protocol for the PPTM problem.*

This theorem is postponed to the appendix.

4.6.2 Security Analysis for Protocol 2

The Inferred Information by the Definition of PPTAM

By the definition of PPTAM in Section 4.1, it is unavoidable for P_i $(i \in I)$ to infer the following, if the colluded parties combine their inputs and outputs:

1) if $T(i,j) \in T1_i$,

   1.1) if $\exists M' \subseteq \{1,...,M\}$, $\forall k \in M'$, $T(i,j)_k$ appears at least 2 times in $A_k^I = \{T(i,j)_k | i \in I, j = 1,...,S\}$, then P_i does not know whether $T(i,j)_k$ is in $A_k^{I'} = \{T(i,j)_k | i \in I', j = 1,...,S\}$.

   1.2) if $\exists M' \subseteq \{1,...,M\}$, $\forall k \in M'$, $T(i,j)_k$ appears only one time in A_k^I, then P_i knows $T(i,j)_k$ appears at least one time in $A_k^{I'}$. We denote these $T(i,j)_k$ in $A_k^{I'}$ as $\mathcal{TA}$ (the cardinality of $\mathcal{TA}$, $|\mathcal{TA}| \leq (N-c)S$).

2) if $T(i,j) \in T0_i$,

2.1) if $\exists M' \subset \{1,...,M\}$, $\forall k \in M'$, $T(i,j)_k$ appears at least 2 times in A_k^I, P_i does not know whether $T(i,j)_k$ is in $A_k^{I'}$.

2.2) if on $M' \subseteq \{1,...,M\}$, $\forall k \in M'$, $T(i,j)_k$ appears only one time in A_k^I, then P_i knows $\exists k' \in M'$, $T(i,j)_{k'}$ is not in $A_{k'}^{I'}$, but does not know the specific k'. We denote these $T(i,j)_{k'}$ as $\mathcal{TA}'$.

The Inferred Information After Participating in Protocol 2

Let $F_k = \prod_{i \in I} f_k^i$, and $G_k = \prod_{i \in I'} f_k^i$. From the view of P_i ($i \in I$), inside each g_l ($l = 1,...,M$), $f_k^{(1)} = [G_k * F_k]^{(1)}$. P_i can do an attack to analyze the coefficients of G_k by combining the inputs and outputs of its colluded parties after participating in Protocol 2. However, in the following lemma, we prove that Protocol 2 is robust against this attack.

Lemma 13 *In Protocol 2, any P_i in the coalition of c ($c \leq N-1$) parties $(P_{i_1},...,P_{i_c})$ can get only the following information:*

1) $\mathcal{TA}$ and $\mathcal{TA}'$ which are the same with those described in Section 4.6.2.1.

2) guessing other roots of G_k ($k = 1,...,M$) than $\mathcal{TA}$ and $\mathcal{TA}'$, after randomly choosing at least 2 numbers.

Proof: We use $g_l(T(i,j))$ to denote the evaluation of $g_l(x_1,...,x_M)$ at $(T(i,j)_1,...,T(i,j)_M)$ for simplification. Firstly, we prove that the c colluded parties are unable to find a unique g_l ($l = 1,...,M$) that fits cS pairs ($T(i,j), g_l(T(i,j))$) ($i \in I$, $j = 1,...,S$). For P_i, $g_l = \sum_{k \in \{1...M\}}(R_{lk} f_k^{(1)} * \sum_{i=1}^N s_{ik})$. $Degree(f_k^{(1)}) = NS-1$, $Degree(s_{ik}) = (N-1)S$. If P_i knows all coefficients of g_l, it can factor each $R_{lk} f_k^{(1)} * \sum_{i=1}^N s_{ik}$ and know all attribute values that appear at least 2 times on all parties. Actually P_i does not know R_{lk}. $f_k^{(1)}$ has $NS-1$ unknown coefficients for P_i, excluding the highest order coefficient ($=1$). $\sum_{i=1}^N s_{ik}$ can be considered as a polynomial $\sum_{j=0}^{(N-1)S} R'_j x_k^j$, and has $(N-1)S+1$ unknown coefficients. Then P_i has totally $(1 + NS - 1 + (N-1)S + 1)M = (2NS - S + 1)M$ unknown coefficients in g_l. P_i must randomly guess $t = (2NS-S+1)M - cS$ coefficients to find all coefficients of g_l, and it is easy to see that $t > NS+1$ for $c \leq N-1$ and $M \geq 1$.

Let $f_k = G_k * F_k$ in which G_k and F_k are polynomials representing inputs of $P_{I'}$ and P_I respectively. $Degree(G_k) = (N-c)S$, $Degree(F_k) = cS$. P_i can notice that $g_l = \sum_{k \in \{1...M\}}(R_{lk} h_k)$ in which $h_k = [G_k * F_k]^{(1)} * (\sum_{i \in I} s_{ik} + \sum_{i' \in I'} s_{i'k})$, $[G_k * F_k]^{(1)} = G_k^{(1)} F_k + G_k F_k^{(1)}$, $\sum_{i \in I} s_{ik}$ is generated by P_I, and $\sum_{i' \in I'} s_{i'k}$ is generated by $P_{I'}$. Then from P_i's view, the undetermined parts of g_l are R_{lk}, G_k and $\sum_{i' \in I'} s_{i'k}$ for $k \in \{1,...,M\}$.

Supposing $T(i,j)_k$ appears m_k times in A_k^I, then P_i knows F_k has a factor of $(x - T(i,j)_k)^{m_k}$. From the values of ($g_1(T(i,j)),...,g_M(T(i,j))$), P_i can analyze the roots of G_k as follows:

1) If ($g_1(T(i,j)),...,g_M(T(i,j))$) $= (0,...,0)$, by Lemma 4, P_i knows $h_k(T(i,j)_k) = 0$ for all $k \in \{1,...,M\}$, and $G_k * F_k$ has a factor $(x - T(i,j)_k)^{n'}$ ($n' \geq 2$), but does not know what n' is.

1.1) if $\exists M' \subseteq \{1,...,M\}$, $\forall k \in M'$, $m_k \geq 2$, then P_i does not know whether G_k has a factor $(x - T(i,j)_k)$. This accords with the case 1.1) in Section 4.6.2.1.

1.2) If $\exists M' \subseteq \{1, ..., M\}$, $\forall k \in M'$, $m_k = 1$, then P_i knows there is a factor $(x - T(i,j)_k)^{n'-1}$ in G_k. These $T(i,j)_k$ are the same with $\mathcal{TA}$.

2) If ($g_1(T(i,j)), ..., g_M(T(i,j))$) $\neq (0, ..., 0)$,

2.1) if $\exists M' \subset \{1, ..., M\}$, $\forall k \in M'$, $m_k \geq 2$, then P_i knows $h_k(T(i,j)_k) = 0$, but does not know whether G_k has a factor $(x - T(i,j)_k)$. This accords with the case 2.1) in Section 4.6.2.1.

2.2) P_i can find out that $\exists M' \subseteq \{1, ..., M\}$, $\forall k \in M'$, $m_k = 1$, then P_i knows there must be some $k' \in M'$, such that $h_{k'}(T(i,j)_{k'}) \neq 0$, but does not know the specific k'. P_i also knows that $G_{k'}$ can not have a factor $(x - T(i,j)_{k'})^d$ $(d \geq 1)$. These $T(i,j)_{k'}$ compose $\mathcal{TA}'$.

From the above analysis, only in the case of 1.2), P_i can know some roots of G_k for $k \in M'$, $M' \subseteq \{1, ..., M\}$. Suppose the number of these roots is C_{1k}. If C_{1k} is rightly $(N - c)S$, P_i knows all roots of G_k, but this seldom happens. P_i may use the roots found in 1.2) to compute the unknown coefficients of G_k in g_l.

In the worst situation, $|M'| = 1$. That is, for $k \in M'$, $g_l = R_{lk}h_k = R_{lk}(G_k * F_k)^{(1)} * (\sum_{i \in I} s_{ik} + \sum_{i' \in I'} s_{i'k})$. P_i does not know R_{lk}. G_k has $(N - c)S$ unknown coefficients excluding the highest order coefficient (=1). $\sum_{i' \in I'} s_{i'k}$ can be considered as a polynomial $\sum_{j=0}^{(N-1)S} R''_j x_k^j$, and has $(N - 1)S + 1$ unknown coefficients. Then P_i has totally $1 + (N - c)S + (N - 1)S + 1$ unknown coefficients in g_l. It knows C_{1k} solutions to this g_l. P_i still needs to guess $t = 1 + (N - c)S + (N - 1)S + 1 - C_{1k}$ coefficients to know all coefficients of g_l. $C_{1k} \leq (N - c)S$, $c \leq N - 1$, then $t \geq 2$.

In a better situation than above, P_i should guess $t > 2$ coefficients to know all coefficients of g_l.

Therefore P_i should randomly select at least two numbers to guess any root of G_k other than $\mathcal{TA}$ and $\mathcal{TA}'$. ■

Theorem 8 *Protocol 2 is a privacy preserving protocol for the PPTAM problem.*

The proof of this theorem is postponed to the appendix.

4.7 Comparisons with Related Work

Many ways have been proposed to improve the efficiency of modular exponentiation (e.g., [3], [90]), and by these results a modular exponentiation generally requires $O(lg\mathcal{N})$ modular multiplications ($\mathcal{N}$ is the modulus). Therefore, modular exponentiation is the major cause of computation cost for Paillier's cryptosystem and our protocols. We compute the total *mod-exp*s (modular exponentiations with modulus $\mathcal{N}^2$) to compare our protocols with others. We will not consider their costs on key generation because they are the same.

4.7.1 Comparisons for Protocol 1

Computation Cost of Protocol 1: According to [33], in the threshold version of Paillier's cryptosystem, each encryption requires 2 mod-exps, the decryption is performed by the combination of all parties, and each party computes 2 mod-exps for the decryption.

Computation 4) of Section 4.3.3 is a basic computation for Protocol 1. Given $degree(f) = m$ and $degree(g) = n$, the cost for $E(f(x) * g(x))$ is $O(mn)$ mod-exps. The j-th ($j \geq 2$) iteration of Step 2 is paralleled on the N parties, but P_1 is responsible for the major cost. Give $degree(F_{1,(j-1)}) = (j-2)S + (j-2)\lceil \frac{S}{N-1} \rceil$, $degree(f_1) = S$, $degree(r_{j1}) = degree(r'_{j1}) = \lceil \frac{S}{N-1} \rceil$, the computation cost of P_1 is $O(\ ((j-1)S + (j-2)\frac{S}{N-1})\ \frac{S}{N-1}\ )$ mod-exps. Thus from $j = 2, ..., N$, the total cost of P_1 in Step 2) is $O(NS^2)$ mod-exps. Furthermore, the cost can be optimized if all parties ensure that the number of semi-honest parties is at most c ($1 \leq c \leq N-1$). For this case, in Step 2.1), P_1 only need send $E(F_{1,(j-1)} * f_1)$ to c arbitrary parties; in Step 2.2), P_j also only need send $E(F_{1,(j-1)} * f_j)$ to c arbitrary parties; At the end of Step 2), P_1 gets $E(F_1) = E(\prod_{j=2}^{N}(f_1 * \sum_{k=1}^{c+1} r_{jk} + f_j * \sum_{k=1}^{c+1} r'_{jk}))$. In this polynomial, P_1 has at least one unknown r_{jk} and one unknown r'_{jk}, which still ensures the security of F_1 against the analysis of P_1's coalition.

Step 3) can be made paralleled with Step 2). Each P_i can compute its $E(F_i)$ without delaying the computation of the others' $E(F_{i'})$ ($i' \neq i$), in the assumption that each party holds N computation platforms. In Step 4) $degree(F_i) = NS$, the cost for the evaluations of S tuples on each party is $O(NS^2)$ mod-exps. In Step 5) the cost for the decryptions of S evaluations is $O(S)$ mod-exps.

In sum the total cost of Protocol 1 for each party is $O(NS^2)$ mod-exps.

Communication Cost of Protocol 1: The length of each encryption is $O(lg\mathcal{N})$ bits. The major communication cost of Protocol 1 is on Step 2) and 3). In the j-th ($j \geq 2$) iteration of Step 2.1), P_1 sends an encrypted polynomial with degree $(j-1)S + (j-2)\lceil \frac{S}{N-1} \rceil$ to the other $N-1$ parties, and all the other parties send back encrypted polynomials with degree $(j-1)S + (j-1)\lceil \frac{S}{N-1} \rceil$. In Step 2.2) a similar round happens. Thus the total bandwidth of Step 2) is $O((N^3S))$ encryptions. Assuming the number of semi-honest parties is at most c ($1 \leq c \leq N-1$), the total bandwidth of Step 2) can be optimized as $O(cN^2S)$ encryptions.

Step 3) iterates Step 2) for $N-1$ rounds, so the total bandwidth of Protocol 1 is $O(cN^3S)$ encryptions, i.e., $O(cN^3Slg\mathcal{N})$ bits.

Costs of Solution D1: The main idea of the private set intersection protocol in [62] is to plus the randomized polynomials representing the data sets and evaluate the resulting polynomial. Their private set union protocol is mainly to multiply the polynomials representing the data sets and evaluate the resulting polynomial. Therefore, Solution D1 (as described in Section 4.2) should firstly know $(f_i * \sum_{k=1}^{N} r_{i'k} + f_{i'} * \sum_{k=1}^{N} r'_{i'k})$ for $T_i \cap T_{i'}$ for $i' = 1, ..., N$, $i' \neq i$, then know $F_i = \prod_{i'=1...N, i' \neq i}(f_i * \sum_{k=1}^{N} r_{i'k} + f_{i'} * \sum_{k=1}^{N} r'_{i'k})$ for $\bigcup_{i'=1...N, i' \neq i}(T_i \cap T_{i'})$, and evaluate it. However, the way to privately compute the encrypted F_i was not provided in [62], and all $r_{i'k}$ and $r'_{i'k}$ are randomly chosen polynomials with degree S.

In this solution, if $F_i(T(i,j)) = 0$, overwhelmingly $T(i,j)$ has a duplicate with some $P_{i'}(i' \in \{1, ..., N\} \setminus \{i\})$. F_i must not be decrypted before evaluations, otherwise by factoring the decrypted F_i, P_i will know how many duplicates of $T(i,j)$ there are on the other parties, thus breach the second privacy requirement in Section 4.1. Because $r_{i'k}$ and $r'_{i'k}$ have the same degree with f_i and f'_i, the major cost of Solution D1 is on the

computation and evaluations of $E(F_i)$.

F_i can be computed following the same way in Protocol 1. Assuming the number of semi-honest parties is c, in the j-th iteration of Step 2), the computation cost of P_1 and bandwidth are $O(jS^2)$ mod-exps and $O(jcSlg\mathcal{N})$ bits. From $j = 2, ..., N$ the total computation cost of P_1 and bandwidth are $O(N^2S^2)$ mod-exps and $O(cN^2Slg\mathcal{N})$ bits. Thus, the total computation cost and bandwidth of Solution D1 can be got in a similar way with Protocol 1 and given in Table 4.3.

Costs of TCTS Protocol in [63]: Before the TCTS protocol in [63] can be employed for the PPTM problem, a process of tuple deduplication should be performed inside each party's database. The TCTS protocol computes the encryption of the polynomial $p = \prod_{i=1}^{N} f_i$, then $\Phi = p * (\sum_{i=1}^{N} r_i) + F * p^{(1)} * (\sum_{i=1}^{N} s_i)$, where each r_i and s_i are polynomials of degree NS which are randomly selected, F is a fixed polynomial of degree 1 which has no common roots with p. The major cost of the TCTS protocol is on the computation of $E(\Phi)$. Given $E(p)$ and any r_i, $degree(p) = NS$, so the cost for computing $E(p * r_i)$ is $O(N^2S^2)$ mod-exps. The communication cost of TCTS can be obtained by the similar way used for Solution D1. The total costs are given in Table 4.3.

Table 4.3: Comparison of solutions for the PPTM problem

	PPTM Protocol of Ours	*Solution D1*	*TCTS Protocol in [63]*
Computation Cost (mod-exps)	$O(NS^2)$	$O(N^2S^2)$	$O(N^2S^2)$
Communication Cost (bits)	$O(cN^3Slg\mathcal{N})$	$O(cN^3Slg\mathcal{N})$	$O(N^3Slg\mathcal{N})$

4.7.2 Practical Considerations and Comparisons for Protocol 1

Suppose there is a moderate-scale application with 5 parties. 4 parties of them may collude, each party has 500 tuples in its database, and each tuple has 5 integer fields, i.e., $N = 5$, $c = 4$, $S = 500$. Each tuple will be at most 160 bits. Suppose $\mathcal{N}$ be 1024 bits. The polynomial f_i has 501 $(= S+1)$ coefficients and each coefficient is at most 1024 bits, so the storage for f_i is about $62K$ bytes. The largest storage is spent on $E(F_i)$, which has 2501 $(= NS + 1)$ encrypted coefficients and need a storage of about $621K$ bytes when $|\mathcal{N}^2| = 2048$.

Mod-exp is a basic computation for the 3 protocols, so we test its running time on a computer with a CPU of 2.8GHz (Pentium 4), and get the following table:

Table 4.4: Running Time of Modular Exponentiation

the length of modulus	*512-bit*	*1024-bit*	*2048-bit*
running time (ms)	0.63	1.22	4.07

If $lg\mathcal{N} = 512 \sim 1024$, i.e., $|\mathcal{N}^2| = 1024 \sim 2048$, in the supposed application, our Protocol 1 can be completed in tens of minutes, but Solution D1 and TCTC protocol need a few hours. The computation time of Protocol 1 has a reduction of 80% in comparison with Solution D1 and TCTS protocol. Given a communication network with a T1 line of 1.5Mbps or a T3 line of 32Mbps, the communication bits of Protocol 1 can be transferred within a few minutes or seconds. Therefore, in comparisons, our Protocol 1 has lower computation time without increasing intolerable bandwidth.

4.7.3 Comparison of Protocol 2 with Solution D2

Computation Cost of Protocol 2 Similarly with Protocol 1, we compare Protocol 2 with Solution D2 in terms of their mod-exps. Protocol 2 can be executed in parallel supposing each party has M computation platforms. In Step 1) P_i computes its $E(f_k^{(1)} * \sum_{i=1}^{N} s_{ik})$ on its k-th platform ($k \in \{1, ..., M\}$). The major time cost of Step 1) is on Step 1.5): given $degree(f_k^{(1)}) = NS - 1$, $degree(s_{ik}) = (N-1)S$, the time cost of computing $E(f_k^{(1)} * s_{ik})$ is $O(N(N-1)S^2)$ mod-exps. In Step 2) the l-th ($l \in \{1, ..., M\}$) platform of P_i holds the l-th column of the matrix $\mathcal{R}_i\mathcal{U}$, computes the l-th element of $E(\mathcal{F}\mathcal{R}_i\mathcal{U})$, exchanges it with other parties, and computes $E(g_l)$ by summing the l-th element of $E(\mathcal{F}\mathcal{R}_i\mathcal{U})$ for $i = 1, ..., N$. The major cost of Step 2) is on the Step 2.3): given $E(\mathcal{F})$ and the l-th column of $\mathcal{R}_i\mathcal{U}$, the time cost of computing the l-th element of $E(\mathcal{F}\mathcal{R}_i\mathcal{U})$ is $O(2NSM)$ mod-exps. In Step 3) the l-th platform evaluates $T(i, j)$ at $E(g_l)$ for $j = 1, ..., S$, so the time cost is $O(2NS^2M)$ mod-exps. Therefore, the total time cost of Protocol 2 can be got by summing the major costs on the 3 steps, i.e., $O((2M + N - 1)NS^2 + 2NMS)$ mod-exps.

Communication Cost of Protocol 2 The major bandwidth of Step 1) is spent on Step 1.5): $degree(f_k^{(1)} * s_{ik}) = (2N-1)S - 1$, so the total transferred encryptions is $O(2(N-1)N^2MS)$. The bandwidth can be optimized if the number of semi-honest parties is c. Then only $c+1$ parties need compute $E(f_k^{(1)} * s_{ik})$ and send it to all other parties, and in the end of 1.6) each P_i gets $E(f_k^{(1)} * \sum_{i=1}^{c+1} s_{ik})$. The bandwidth of Step 1) becomes $O(2(c+1)N(N-1)MS)$ encryptions. The bandwidth of Step 2) is on Step 2.4), each element of $E(\mathcal{F}\mathcal{R}_i\mathcal{U})$ is composed of M polynomials with degrees of $(2N-1)S-1$, then each P_i need transfer $O(2N(N-1)MS)$ encryptions to the others, and the total bandwidth is $O(2(N-1)N^2MS)$ encryptions. In Step 3) the bandwidth of each decryption is $O(N)$, so the total bandwidth of Step 3) is $O(N^2MS)$. In sum, the total bandwidth of Protocol 2 is $O((N-1)N^2MS)$ encryptions, i.e., $O((N-1)N^2MSlg\mathcal{N})$ bits.

Costs of Solution D2 The main idea of Solution D2 (as described in Section 4.2) is to: 1) set $t = 2$ in the threshold set union protocol of [62], compute the encrypted forms of $f_k^{(1)}$ ($k \in \{1, ..., M\}$), and randomize them to be $f_k^{(1)}F_k\sum_{i=1}^{N} s_k^i + f_k\sum_{i=1}^{N} r_k^i$, in which F_k is a fixed polynomial of degree 1 which has no same roots with f_k. s_k^i and r_k^i are randomly selected polynomials of degree NS over polynomial ring $R[x_k]$. 2) following the idea of the private set intersection protocol in [62], an encrypted multivariate polynomial is constructed to compute $TN_1^i \cap TN_2^i \cap ... \cap TN_M^i (= T1_i)$: $G(x_1, ..., x_M) = \sum_{k=1}^{M}\left(f_k^{(1)}F_k\sum_{i=1}^{N} s_k^i + f_k\sum_{i=1}^{N} r_k^i\right) * (\sum_{i=1}^{N} r'^i_k)$. r'^i_k is a polynomial of degree $2NS$ randomly selected by P_i over the polynomial ring $R[x_k]$. If $T(i, j) \in T1_i$, the evaluation of $G(x_1, ..., x_M)$ at $T(i, j)$ is 0. If the evaluation is 0, overwhelmingly $f_k^{(1)}(T(i, j)_k) = 0$ for $k = 1, ..., M$, then $T(i, j) \in T1_i$.

What's more, $G(x_1, ..., x_M)$ must not be decrypted before it is evaluated, otherwise a semi-honest party will know all roots of $f_k^{(1)}$ by factoring $(f_k^{(1)}F_k\sum_{i=1}^{N} s_k^i + f_k\sum_{i=1}^{N} r_k^i) * (\sum_{i=1}^{N} r'^i_k)$, then it will know the specific appearance times of the attribute values on all parties if these values appear at least n times, thus breach the second privacy requirement in Section 4.1.

Solution D2 has a same step with Step 1) of Protocol 2. Then P_i need $O(N^2S^2)$ mod-exps to compute $f_k\sum_{i=1}^{N} r_k^i$, $O(4N^2S^2)$ mod-exps to compute $(f_k^{(1)}F_k\sum_{i=1}^{N} s_k^i + f_k\sum_{i=1}^{N} r_k^i) * (\sum_{i=1}^{N} r'^i_k)$, on its k-th platform ($k \in \{1, ..., M\}$). Finally P_i need $O(4NMS^2)$

mod-exps to evaluate $G(x_1, ..., x_M)$ at $T(i, j)$ for $j = 1, ..., S$. Totally the time cost of P_i is $O(N(N-1)S^2+5N^2S^2+4NMS^2) = O((6N+4M-1)NS^2)$. The major bandwidth of Solution D2 is on the following step: each P_i transfers $(f_k^{(1)}F_k \sum_{i=1}^{N} s_k^i + f_k \sum_{i=1}^{N} r_k^i) * r'^i_k$ to all the other parties. So the total bandwidth is $O(4(N-1)N^2MS)$ encryptions, i.e., $O(4(N-1)N^2MS lg\mathcal{N})$ bits.

We compare Protocol 2 with Solution D2 in Table 4.5.

Table 4.5: Comparison of solutions for the PPTAM problem

	PPTAM Protocol of Ours	*Solution D2*
Computation Cost (mod-exps)	$O((2M+N-1)NS^2+2NMS)$	$O((6N+4M-1)NS^2)$
Communication Cost (bits)	$O((N-1)N^2MS lg\mathcal{N})$	$O(4(N-1)N^2MS lg\mathcal{N})$

4.7.4 Practical Considerations and Comparisons for Protocol 2

We assume there is a same moderate-scale application with Section 4.7.2, i.e., $N = 5$, $M = 5$, $S = 500$. Given the running times of mod-exps in Table 4.5, when $lg\mathcal{N} = 512 \sim 1024$, i.e., $|\mathcal{N}^2| = 1024 \sim 2048$, the computation cost of Protocol 2 is about 5 hours, and has a reduction of about 71.3% in comparison with Solution D2. The communication cost of Protocol 2 is about 2.56×10^8 bits, which will be transferred in a few minutes or seconds within a network linked by a T1 or T3 line. The communication cost of Protocol 2 has a reduction of about 75% in comparison with Solution D2.

The largest storage is spent on the polynomial $E(g_l)$ on the l-th platform of P_i. $E(g_l)$ has $(2N-1)SM$ encrypted coefficients. When $|\mathcal{N}^2| = 1024 \sim 2048$, the storage is about $2.6 \sim 5.2M$ bytes.

4.8 Concluding Remarks and Open Problems

We present protocols respectively for the problem of privacy preserving tuple matching (PPTM) among N parties, and the problem of privacy preserving threshold attributes matching (PPTAM) among N parties. Solutions can also be derived directly from the techniques in [62] and [63]. In comparisons, our protocol for PPTM has a lower computation cost without increasing intolerable communication cost, and our protocol for PPTAM achieves lower computation and communication costs. Both of our protocols are proved to be secure in the semi-honest model, which is the same level of security with the derived solutions from [62] and [63].

Experiments in a moderate-scale application with Pentium 4 platforms also show that the response time of Protocol 1 is tens of minutes, and the time of Protocol 2 is a few hours. As pointed out in [71], for long-term security a 1024-bit modulus should be considered. There are also applications which are large-scale, e.g., with thousands of tuples in each party's database, or need larger key size to ensure stronger security. For these applications, platforms with higher computation capability will be required to get reasonable response time.

Our two proposed protocols can be extended to be secure in the malicious model where a malicious party may arbitrarily substitute its inputs or intermediate computation, and

may quit the protocol at any time it gets desired results. By [48], assuming there is a malicious and PPT bounded adversary who controls arbitrary number of parties, the main idea about the extensions is to add zero-knowledge proofs on each step, then a prover in the zero-knowledge proof must behave at most like a semi-honest one, otherwise its cheating will be detected by the verifier. However, zero-knowledge proofs may cause too much overhead. For example, in computation 4) of Section 4.3.3, to prove $E(g(x))$ is correctly multiplied by a polynomial $f(x)$, the prover should prove that the coefficient $E(c_k)$ is correctly computed as $E(a_0b_k + ... + a_kb_0)$ where he knows a_0, ..., a_k. The proof can be based on the proof of correct multiplication in [15], which need $O(1)$ mod-exps. Then each proof for correct $E(c_k)$ is $O(k)$ mod-exps, and the total cost for proof of correct $E(g(x)f(x))$ is $O(mn)$ mod-exps. By this proof, in Step 1.5) of Protocol 2, the proof of correct $E(f_k^{(1)} * s_{ik})$ has a cost of $O(N(N-1)S^2)$ mod-exps, which need a few hours to be completed in a moderate-scale application. Therefore, how to make the extended protocols efficient in the malicious model is a challenging work for the future.

4.9 Appendix

Theorem 7: *Protocol 1 is a privacy preserving protocol for the PPTM problem.*

Proof By the definition of PPTM, we actually should compute a multi-party function f: $f(T_1, ..., T_N) = f(\overline{T}) = \{PPTM(T(i,j))|T(i,j) \in T_i, i = 1, ..., N, j = 1, ..., S\}$, with the i-th element $f_i(\overline{T}) = \{PPTM(T(i,j))|T(i,j) \in T_i, j = 1, ..., S\}$ for the party P_i, where $PPTM(T(i,j)) = 1$ if $T(i,j) \in T_i \cap (\bigcup_{i'=1...N, i' \neq i} T_{i'})$, and $PPTM(T(i,j)) = 0$ if $T(i,j) \overline{\in} T_i \cap (\bigcup_{i'=1...N, i' \neq i} T_{i'})$.

Given any coalition of c ($c \leq N-1$) semi-honest parties indexed by $I = \{i_1, ..., i_c\}$, their views after participating in Protocol 1 are denoted by $VIEW_I^{\Pi}(\overline{T}) = (I, VIEW_{i_1}^{\Pi}(\overline{T}), ..., VIEW_{i_c}^{\Pi}(\overline{T}))$. We also let $f_I(\overline{T}) = (f_{i_1}(\overline{T}), ..., f_{i_c}(\overline{T}))$. From the definition in Section 4.3.1, we have to show that there exists a PPT algorithm $\mathcal{S}$ such that $\mathcal{S}(I, (T_{i_1}, ..., T_{i_c}), f_I(\overline{T}))$ and $VIEW_I^{\Pi}(\overline{T})$ are computationally indistinguishable.

$VIEW_I^{\Pi}(\overline{T}) = \{V_1, V_2, V_3, V_4\}$: 1) V_1 is $I = \{i_1, ..., i_c\}$. 2)V_2 are $T_{i_1}, ..., T_{i_c}$. 3)V_3 are $E(F_i)$ and the intermediate encryptions received by P_I. 4)V_4 are $F_{i_t}(T(i_t, j))$ for any $i_t \in I$.

With the above views, the coalition can do the following two types of analysis:

1) *Cryptanalysis on* (V_1, V_2, V_3): Due to the semantic security of the threshold Paillier's cryptosystem, P_i can not gain extra information from the encryptions in V_3. That is, supposing V_3 has s encryptions, with only negligible probability, P_i can distinguish V_3 and $\mathcal{ER}_1 = (E(r_1), ..., E(r_s))$ by randomly choosing $\mathcal{R}_1 = (r_1, ..., r_s)$ over the plaintext space of Paillier's cryptosystem. Thus, $(V_1, V_2, V_3) \equiv^c (V_1, V_2, \mathcal{R}_1, \mathcal{ER}_1)$.

2) *Roots analysis on* (V_1, V_2, V_4): From Lemma 9, $V_4 = (\mathcal{A}, \mathcal{T}, \mathcal{T}', \mathcal{R}_2)$. $\mathcal{A} = \{a_{i_t}^j | i_t \in \{i_1, ..., i_c\}, j = 1, ..., S\}$ in which $a_{i_t}^j = 1$ if $F_{i_t}(T(i_t, j)) = 0$, and $a_{i_t}^j = 0$ otherwise. $\mathcal{R}_2 = \{R_i | i = 1, ..., t\}$, in which R_i is a random number guessed by P_i, $t \geq 1$.

In sum, $VIEW_I^{\Pi}(\overline{T}) \equiv^c (V_1, V_2, \mathcal{R}_1, \mathcal{ER}_1, \mathcal{A}, \mathcal{T}, \mathcal{T}', \mathcal{R}_2)$.

Let $\mathcal{R}_1' = \{r_i' | i = 1, ..., s\}$, $\mathcal{R}_2' = \{R_i' | i = 1, ..., t\}$ are randomly chosen by P_I, and $\mathcal{ER}_1'$ are the encryptions of the sequence in $\mathcal{R}_1'$, then we can define $\mathcal{S}(I, (T_{i_1}, ..., T_{i_c}), f_I(\overline{T})) =$

$(I, (T_{i_1}, ..., T_{i_c}), f_I(\overline{T}), \mathcal{R}'_1, \mathcal{ER}'_1, \mathcal{R}'_2)$. $f_I(\overline{T}) = (\mathcal{A}, \mathcal{T}, \mathcal{T}')$ by the analysis in Section 4.4.3.1. Then $\mathcal{S}(I, (T_{i_1}, ..., T_{i_c}), f_I(\overline{T})) = (I, (T_{i_1}, ..., T_{i_c}), \mathcal{A}, \mathcal{T}, \mathcal{T}', \mathcal{R}'_1, \mathcal{ER}'_1, \mathcal{R}'_2) \equiv^c (V_1, V_2, \mathcal{A}, \mathcal{T}, \mathcal{T}', \mathcal{R}_1, \mathcal{ER}_1, \mathcal{R}_2) \equiv^c VIEW_I^{\Pi}(\overline{T})$. Then Protocol 1 privately computes PPTM against the coalition of any c ($c \leq N-1$) semi-honest parties. ■

Theorem 8: *Protocol 2 is a privacy preserving protocol for the PPTAM problem.*

Proof: By the definition of PPTAM, we should compute a multi-party function f: $f(T_1, ..., T_N) = f(\overline{T}) = \{PPTAM(T(i,j))|i = 1, ..., N, j = 1, ..., S\}$, with the i-th element $f_i(\overline{T}) = \{PPTAM(T(i,j))|j = 1, ..., S\}$, where $PPTAM(T(i,j)) = 1$ if $T(i,j) \in T1_i$, and $PPTAM(T(i,j)) = 0$ if $T(i,j) \in T0_i$.

Given any coalition of c ($c \leq N-1$) semi-honest parties indexed by $I = \{i_1, ..., i_c\}$, their views after participating in Protocol 2 are denoted by $VIEW_I^{\Pi}(\overline{T}) = (I, VIEW_{i_1}^{\Pi}(\overline{T}), ..., VIEW_{i_c}^{\Pi}(\overline{T}))$. We also let $f_I(\overline{T}) = (f_{i_1}(\overline{T}), ..., f_{i_c}(\overline{T}))$. We have to show that there exists a PPT algorithm $\mathcal{S}$ such that $\mathcal{S}(I, (T_{i_1}, ..., T_{i_c}), f_I(\overline{T}))$ and $VIEW_I^{\Pi}(\overline{T})$ are computationally indistinguishable.

$VIEW_I^{\Pi}(\overline{T})$ includes the following: 1) V_1: $I = \{i_1, ..., i_c\}$, 2) V_2: $T_{i_1}, ..., T_{i_c}$, 3) V_3: $E(\mathcal{G})$ and intermediate encryptions received by P_I, 4) V_4: $\mathcal{U}$, $\mathcal{R}_{i_t}$ for any $i_t \in I$, 5) V_5:$Y(i_t, 1), ..., Y(i_t, S)$ for any $i_t \in I$.

With the above views, the coalition can do the following two types of analysis:

1) *Cryptanalysis on the encryptions*: Due to the semantic security of the threshold Paillier's cryptosystem, P_i can not gain extra information from the encryptions in V_3 combined with V_1, V_2, and V_4. Supposing V_3 has s encryptions, with only negligible probability, P_i can distinguish V_3 and $\mathcal{ERS}_1 = (E(r_1), ..., E(r_s))$ by randomly choosing $\mathcal{RS}_1 = (r_1, ..., r_s)$ over $\mathbb{Z}_{\mathcal{N}}$. Thus $(V_1, V_2, V_3, V_4) \equiv^c (V_1, V_2, \mathcal{RS}_1, \mathcal{ERS}_1)$.

2) *Roots analysis on the multivariate polynomials*: By Lemma 11, with (V_1, V_2, V_5), P_i can get nothing more than $(\mathcal{A}, \mathcal{TA}, \mathcal{TA}', \mathcal{RS}_2)$. $\mathcal{A} = \{a_{i_t}^j | i_t \in \{i_1, ..., i_c\}, j = 1, ..., S\}$ in which $a_{i_t}^j = 1$ if $Y(i,j) = (0, 0, ..., 0)$, and $a_{i_t}^j = 0$ if $Y(i,j) \neq (0, 0, ..., 0)$. $\mathcal{RS}_2 = \{R_i | i = 1, ..., t\}$, in which R_i is a random number guessed by P_i, $t \geq 2$.

In sum, $VIEW_I^{\Pi}(\overline{T}) \equiv^c (V_1, V_2, \mathcal{A}, \mathcal{TA}, \mathcal{TA}', \mathcal{RS}_1, \mathcal{ERS}_1, \mathcal{RS}_2)$.

From the analysis in Section 4.5.3.1, $\mathcal{S}$ can also find $(\mathcal{A}, \mathcal{TA}, \mathcal{TA}')$. $\mathcal{S}$ can randomly choose $\mathcal{RS}'_1 = \{r'_i | i = 1, ..., s\}$, $\mathcal{RS}'_2 = \{R'_i | i = 1, ..., t\}$ over $\mathbb{Z}_{\mathcal{N}}$, and compute $\mathcal{ERS}'_1$ which are the encryptions of the sequence in $\mathcal{RS}'_1$. Then $\mathcal{S}(I, (T_{i_1}, ..., T_{i_c}), f_I(\overline{T})) = (I, (T_{i_1}, ..., T_{i_c}), \mathcal{A}, \mathcal{TA}, \mathcal{TA}', \mathcal{RS}'_1, \mathcal{ERS}'_1, \mathcal{RS}'_2) \equiv^c VIEW_I^{\Pi}(\overline{T})$. Therefore, Protocol 2 privately computes PPTAM against the coalition of any c ($c \leq N-1$) semi-honest parties. ■

Chapter 5

Privacy State Test in Wireless Sensor Networks

5.1 Problem Background

The term of "ubiquitous computing" was coined by Mark Weiser ([91]) in which computers will vanish into the background and people can access information anytime and anywhere. Sensor networks can be one of the most critical technologies to realize such ubiquitous tasks because large scales of small sensor nodes can be deployed in many physical phenomena and transmit time series of the sensed phenomena to central nodes where computations are performed and data are fused ([89]). A case in point is the petrel habitat monitoring ([88]) in which 32 nodes were deployed on a small island of Maine State of USA and useful live data travels thousands of miles to the orbiting satellite, and then down to the service provider in Washington and Berkeley. Besides, sensor networks may consist of many different types of sensors such as seismic, magnetic, thermal, visual, infrared, acoustic and radar which can monitor a wide variety of ambient conditions ([2]).

It can be envisioned that in the near future, we would live in a world of pervasive sensor networks, where one certain area may be pervasively deployed with various types of sensor networks. As a result, many security issues must be brought to the forefront of the deployment of pervasive sensor networks. [77] has listed the security challenges including key establishment, secrecy, authentication, privacy, robustness to denial-of-service attacks, secure routing, and node capture.

Privacy is the major concern of this chapter. Under the circumstances of pervasive sensor networks, different types of sensor networks may be deployed for different purposes, so a person's privacy claims are dynamic. He may want to publish his information to the location-aware sensors to get his own position, but when he steps into a voice sensing network, he may think it intrusive for others to know what he is talking about. Consequently, it is better to provide a general scheme to address all of these privacy concerns and make the person aware of whether he has been under some observations. We name the person "originator", and his privacy "originator privacy" in this chapter. However, it is another currency that sensor nodes are becoming physically invisible (Smart dust in [59], for example) or should be hidden for military or commercial reasons. We name this kind of privacy "sensing area privacy" in this chapter. How, then, could an originator know the state of his privacy when the sensing area is invisible?

Our Contributions : In this chapter, our major contributions include the following:

1) We explore the issues of sensing area privacy and originator privacy. The two issues conflict with each other, but are all critical concerns in practical applications. To our knowledge, we are the first to address them simultaneously.

2) We provide a solution for the originator to test his privacy state in the case of sensing area privacy, and then judge whether he is infringed by his own initiatives. Our solution is based on the secure two-party point-inclusion problem. To our knowledge, our solution is the first to guarantee the two contradictive privacy concerns without tampering with any of them.

The chapter is organized as following: Section 5.2 introduces the issues and terminologies about sensing area privacy and originator privacy; Section 5.3 outlines the related works; Section 5.4 describes the definition and protocol about the secure two-party point-inclusion problem; Section 5.5 presents the specific scheme to test privacy state in pervasive sensor networks; Section 5.6 gives an evaluation on the scheme; Section 5.7 concludes the chapter with some directions of future work.

5.2 Privacy Issues in Pervasive Sensor Networks

Many definitions on privacy in social science have been reviewed in [72] and an information and communication-based concept was also illustrated. Specifically, privacy is the claim of the manner and extent to which persons can control how information about them is: (1)collected; (2) retained and/or maintained; (3) used; and (4) communicated, disclosed or shared. In pervasive sensor networks, privacy has dual issues in that both sensors and people sensed have concerns of their own privacy.

Originator Privacy : In this chapter we refer to the people who are to be sensed or have been sensed by sensors as *originators*. we give a general term, *originator privacy*, on all of their privacy concerns including individual location, voice, motion, etc.

Generally, the originator privacy has two states: *alert* and *leisure*. When the originator knows that his privacy is on the alert state, he can decide whether his privacy has been infringed, and then whether to protect his privacy. The decision is a subjective thing of the originator. The state of privacy can also fall into other two phases: *infringed* and *not-infringed*. Logically, the alert state contains the whole infringed state and some not-infringed state, whereas the leisure state is contained in the not-infringed state. The relationship of the four states is shown in Fig 5.1. In this chapter we will use the alert and leisure states to test the originator privacy.

Sensing Area Privacy : The physical area where one type of sensor networks is deployed to perform one kind of certain functionality is named *sensing area*. Sensing area may also have privacy. Its lines and vertices shouldn't be guessed by an adversary in order to prevent the sensors from being destroyed. It may also be a secrecy of one company which does not want its competitors to learn the details of the deployment area.

In this chapter, we assume that a sensor network is deployed on a 2-dimensional reference frame, and the coordinates of its vertices for the sensing area have been known after the deployment. The follows are the notations we will use for the sensing area:

- SA: sensing area;

- $\{V_i | i = 1, ..., n\}$: n vertices of SA;

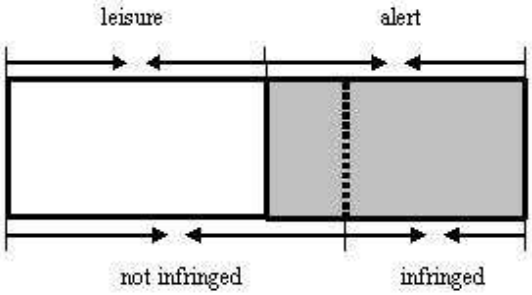

Figure 5.1: *Relationship of four privacy states*

- (x_i, y_i): the coordinate position of the vertex V_i on the 2-dimensional reference frame;
- $\overline{V_iV_{i+1}}$: the line from the vertex V_i to V_{i+1};
- $f_i(x, y)$: the linear equation for the line $\overline{V_iV_{i+1}}$ on the 2-dimensional reference frame.

It is easy to get $f_i(x, y)$ from the coordinates (x_i, y_i) and (x_{i+1}, y_{i+1}) as follows:

$$f_i(x, y) = \frac{y - y_i}{y_{i+1} - y_i} - \frac{x - x_i}{x_{i+1} - x_i} = 0$$

In Fig 5.2, we give an example of sensing area with 5 vertices deployed in one 2-dimensional reference frame.

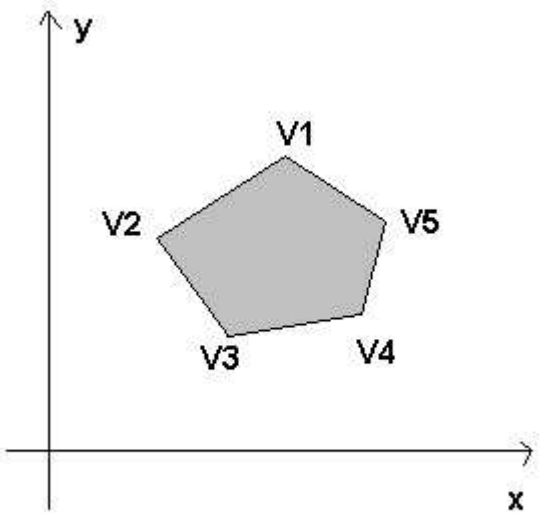

Figure 5.2: *An example of sensing area.*

Two originators and sensing areas of three sensor networks are shown in Fig 5.3. It may be an illustration about the pervasive sensor networks in, say, a park, and the three sensor networks are respectively for surveillance of image, voice, and electromagnetic wave. P_1's privacy is on the alert state, whereas P_2 is on the leisure state.

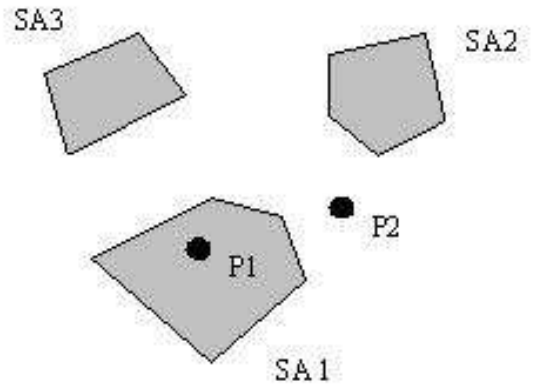

Figure 5.3: *An example of pervasive sensor networks. The deep color parts are sensing areas (SA) of the three sensor networks.* P_1 *and* P_2 *are two originators.*

5.3 Related Work

5.3.1 Key Management Schemes in Wireless Sensor Networks

Many key management schemes have been proposed for *wireless sensor networks* (WSN) to protect *data confidentiality*, which protects the sensitive transmitted data from passive attacks, such as eavesdropping ([69]). Data confidentiality is a vital security issue in a hostile environment, where the wireless channel is vulnerable to eavesdropping. However, in the case that the originators' privacy should also be respected, only key management schemes won't meet the two contradictive privacy requirements from originators and WSN. We give a brief summary on the related work of key management schemes for a clear illustration of their functionalities.

Network – wise Key Distribution : A network-wise key provides end-to-end privacy between the sink and all the sensor nodes. Network-wise key distribution schemes include the master key based and public key based solution:

1) *Master key based solution:* In [13] and [45], it is assumed that the sensor nodes share a common secret key K with the sink, but the aggregator nodes haven't this key. Modular addition and Domingo-Ferrer's scheme ([20]) are used respectively by them to encrypt data, and both of them are additive homomorphic. The limitation is that the whole network will be compromised if K on one sensor node is compromised.

2) *Public key based solution:* In [73], each sensor node uses the public key of the base station to encrypt its reading employing some homomorphic public key encryption schemes. The base station is assumed to have strong reliability so that it is not easy to be compromised. The public key encryption schemes are constructed on elliptic curves in [73], but computation requirement in encryption is still high for the sensor nodes.

Group – wise Key Distribution Schemes Group-wise Key Distribution Schemes are mainly used for hierarchical WSN, where the whole network is hierarchically composed of base stations, cluster heads and sensor nodes. There are two types of distributions:

1) *Symmetric group-wise key distribution:* In [7], a symmetric key can be generated among t nodes by evaluating a symmetric multivariate polynomial $P(x_1, ..., x_t)$ at each node.

2) *Asymmetric group-wise key distribution:* In [70], the memory of each sensor node is pre-loaded with the ECC (elliptic curve cryptography) domain parameters. After deployment, each sensor will compute its EC-public/private key pair and broadcast its public key to all nodes within the cluster. According to their comparisons, the computation complexity of ECC is lower than DSA/RSA cryptosystem, but higher than the symmetric cryptosystem.

Pair − wise Key Distribution Schemes The common way for pair-wise key distribution is key pre-distribution, i.e., keys are stored in sensors before sensors are deployed. After the deployment, each sensor establishes a secret link with its neighbor using a common pair-wise key which has been stored in it. Key connectivity, the probability of one sensor node finds a common key with its neighbor, is an important factor to be considered in the pair-wise key distribution schemes.

1) *Master key based solution:* A simple solution is to store a *master* key in all the sensor nodes ([39]). After they are deployed, each pair of sensor nodes uses this master key to achieve a new pair-wise key. This scheme has low resilience because the compromising of one node will lead to the compromising of the whole network.

2) *Pair-wise key pre-distribution solution:* There is another straightforward solution in which each sensor node stores $N-1$ secret pair-wise keys, each of them is known only to this sensor node and one of the other $N-1$ sensor nodes. This solution has good resilience but is impractical because a sensor node has limited storage and the size of the network (N) could be very large. What's more, this solution isn't scalable to accept new nodes after the deployment of the network because the deployed nodes may haven't the keys of the new node.

3) *Random key pre-distribution solutions:* A basic random key pre-distribution scheme is proposed in [26]: in the *key-predistribution phase*, each sensor node receives a random subset of k keys from a large key pool of K keys. In the *shared-key discovery phase*, to agree on a key for communication, two nodes find one common key within their subsets and use this key as their shared secret key. The probability of key share among two sensor nodes is $\frac{((K-k)!)^2}{(K-2k)!K!}$. In the *path-key establishment phase*, any pair of nodes (i, j) can securely establish a pair-wise key $K_{i,j}$ through a path $i, v_1, ..., v_n, j$, ordinally by sending $E_{K_{i,v_1}}(K_{i,j}), E_{K_{v_1,v_2}}(K_{i,j})..., E_{K_{v_n,j}}(K_{i,j})$. This scheme is improved in [14]: a random set of $(N-1)p$ $(0 < p < 1)$ pair-wise keys is stored in each sensor node. The key connectivity becomes p because with probability p two nodes can be connected. The memory required for storing keys is decreased and good resilience is kept.

4) *Key pre-distribution schemes with deployment knowledge:* A location-based scheme is proposed in [68] to improve the work in [26]: it assumes that each sensor node has an expected location that can be predicted. Then each sensor is preloaded with the pair-wise keys of its c closest neighbors. This solution has low memory usage but good connectivity. Another work in [23] divides sensor nodes into $t \times n$ groups,

and deploys sensors in each group by Gaussian distribution. Compared with [26], key connectivity is improved while keeping good resilience.

5) *Other solutions:* There are also a few key pre-distribution schemes based on other techniques. The scheme in [11] is based on block design in combinatorial design theory. In [22], each pair of nodes can calculate corresponding field of the key matrix and use it as the pair-wise key. The scheme in [42] uses the evaluation of symmetric polynomial $P(x, y)$ ($P(x, y) = P(y, x)$) at the ID of each nodes pair (i, j) to get a pair-wise key $K_{i,j} = P(i, j)$.

5.3.2 Related Work for Protecting Originator's Location Privacy

Originators' location privacy has also been addressed in some work. [51] analyzed the feasibility of anonymizing location information for location-based services in an automotive telematics environment. [5] independently evaluated anonymity techniques for an indoor location system based on the Active Bat. The Cricket Location-Support System [78] incorporates location privacy concern by a design in which device location information is initially only known to the devices themselves. The owner can then conceivably decide to whom this data should be released.

To our knowledge, little work has been done to address the originator privacy when its concern is dynamic under the circumstances of pervasive sensor networks, especially when the sensing area privacy should also been considered. Our scheme is based on the protocol of secure two-party point-inclusion problem [4]. [24] has discussed one military scenario in which such a point-inclusion problem could be used. We apply the point-inclusion protocol to test the originator privacy state in pervasive sensor networks. The point-inclusion protocol will use the protocol for secure two-party vector dominance as a building block, which can be substituted by our STVD protocol in Chapter 2.

5.4 Fundamental Problems of Protocols

The general secure two-party computation problem is to compute one function $f(a, b)$ on multiple parties who are the private owners of a and b respectively, without disclosing anything unintended to each other. The general secure two-party computation problem is solvable using circuit evaluation protocols, but Goldreich has pointed out that solutions derived from these general results can be impractical to solve specific problems[47].

The secure two-party point-inclusion problem can be stated as follows: Alice is the owner of a point p, and Bob is the owner of a polygon P. They merely want to determine whether p is inside P, without revealing to each other any information about the relative position of p and P, such as whether p is approximately at the northeast part of P, or whether p is close to one of the vertices of P, etc.

5.4.1 Two Fundamental Problems

An efficient protocol to solve the secure two-party point-inclusion problem can be based on the secure two-party scalar product protocol and vector dominance protocol.

The secure two-party scalar product problem: Alice has a vector $X = (x_1, ..., x_n)$ and Bob has a vector $Y = (y_1, ..., y_n)$. Alice (not Bob) is to get the result of $u = X \cdot Y + v$ where v is a random scalar known to Bob only. One solution of this problem is from [4] and the main idea is as follows:

1) On Alice, vector X is divided into m random vectors $V_1, ..., V_m$ of which it is the sum ($X = \sum_{i=1}^{m} V_i$).

2) For each V_i, Alice sends p vectors $\{H_1, ..., H_p\}$ to Bob, only one of which equals V_i.

3) Without knowing which one is V_i, Bob then computes the scalar products between Y and each of these p vectors, i.e., $Z_{j,i}$ for $j = 1, ..., p$ as follows:

$$Z_{j,i} = H_j \cdot Y + r_i.$$

in which r_j is a random number generated by Bob and $\sum_{i=1}^{m} r_i = v$.

4) Alice uses the 1-out-of-p oblivious transfer protocol ([47]) to get $V_i \cdot Y + r_i$ back from Bob.

5) After Alice gets $V_i \cdot Y + r_i$ for $i = 1, ..., n$, she can compute

$$\sum_{i=1}^{m} (V_i \cdot Y + r_i) = X \cdot Y + v.$$

The secure two-party vector dominance (STVD) problem: Let $A = (a_1, ..., a_n)$ and $B = (b_1, ..., b_n)$. If for all $i = 1, ..., n$ we have $a_i \geq b_i$, we say that A dominates B. Suppose Alice has A and Bob has B. Alice wants to know whether A dominates B. Note in the case where A does not dominate B, neither Alice nor Bob should learn the relative ordering of any individual a_i, b_i pair, for example, whether $a_i < b_i$ or not.

In Chapter 2 we have proposed an STVD protocol which is more efficient than the related work for the same problem in [4] and [55]. We give the main idea of our STVD protocol to show it can be embedded into a solution for the secure two-party point-inclusion problem as a building block, without any revisions. The detailed executions of the protocol can be found in Chapter 2.

1) For $i = 1, ..., n$, Alice and Bob do random-zero transformations on a_i and b_i respectively. Alice gets a vector $\mathcal{R}_i = (r_{K0}, r_{K1}, ..., r_{10}, r_{11})$, Bob gets K vectors $\mathcal{R}'_{ij} = (r'_{K0}, r'_{K1}, ..., r'_{10}, r'_{11})$ for $j = 1, ..., K$.

2) For $i = 1, ..., n$, Alice and Bob privately compute the product of scalar products, i.e.,

$$E(\prod_{j=1}^{K} (\mathcal{R}_i \cdot \mathcal{R}'_{ij}))$$

3) Alice and Bob compute

$$E(\mathcal{R}) = E(\prod_{j=1}^{K} (\mathcal{R}_1 \cdot \mathcal{R}'_{1j})) \odot \cdots \odot E(\prod_{j=1}^{K} (\mathcal{R}_n \cdot \mathcal{R}'_{nj}))$$

$$= E(\ \sum_{i=1}^{n} (\ \prod_{j=1}^{K} (\mathcal{R}_i \cdot \mathcal{R}'_{ij})\ )\ )$$

4) Alice and Bob cooperatively decrypt $E(\mathcal{R})$. If the decryption $g^{\mathcal{R}} = 1$, they determine that A dominates B, otherwise, they determine that A doesn't dominate B.

5.4.2 Secure Two-party Point-Inclusion Protocol

One $O(n)$ solution on secure two-party point-inclusion problem is firstly to divide edges of the polygon into two boundaries by a diagonal between the leftmost and rightmost vertex. If the point is below all the edges of the upper boundary and above all the edges of the lower boundary, then the point is inside the polygon, otherwise it is outside.

Protocol 1 : *The Protocol for the secure two-party point-inclusion problem:*

Input : Alice has a point $p : (\alpha, \beta)$ and Bob has a polygon with n vertices: $\{V_i | V_i = (x_i, y_i), i = 1, ..., n\}$.

Output : Both of them know whether p is inside the polygon, but Alice doesn't know any geometric information about the polygon, and Bob doesn't know the position of p.

Steps :

1) Bob computes linear equations of each edge of his polygon, by the methods in Section 5.2, and gets $f_i(x, y) = 0$ for the edge $\overline{V_i V_{i+1}}$ $(i = 1, ..., n)$.

2) By comparing the coordinates of the vertices, Bob finds the leftmost and the rightmost vertex. Suppose L is the diagonal between the leftmost and rightmost vertex. All vertices are divided into two sets:

$$\begin{cases} \textit{the upper set } VU = \{i | V_i \textit{ is above } L\} \\ \textit{the lower set } VL = \{i | V_i \textit{ is below } L\} \end{cases}$$

3) Alice and Bob use the secure two-party scalar product protocol to compute u_i for $\overline{V_i V_{i+1}}$ and $i \in VU$ as follows:

$$u_i = -f_i(\alpha, \beta) + v_i,$$

and compute u_i for $\overline{V_i V_{i+1}}$ and $i \in VL$ as follows:

$$u_i = f_i(\alpha, \beta) + v_i.$$

According to the scalar product protocol, Alice will get $(u_1, ..., u_n)$ and Bob will get $(v_1, ..., v_n)$. Bob will learn nothing about u_i and (α, β); Alice will learn nothing about v_i and the function $f_i(x, y)$.

4) Alice and Bob use the secure two-party vector dominance protocol to find out whether vector $A = (u_1, ..., u_n)$ dominates $B = (v_1, ..., v_n)$. According to the vector dominance protocol, if A does not dominate B, no other information is disclosed.

5) If $A = (u_1, ..., u_n)$ dominates $B = (v_1, ..., v_n)$, then the point $p = (\alpha, \beta)$ is inside the polygon; otherwise, the point is outside (or on the edge) of the polygon.

5.5 The Scheme to Test Privacy State

Section 4 has given a solution to address the relationship between a point and a polygon. The solution can also be suitable for describing the geographic relationship between originators and sensor networks.

5.5.1 Who's Alice

Every originator who is to be sensed or has been sensed can act as Alice in secure two-party point-inclusion problem. He has a point p which donates his current position, and he wants to judge whether he has stayed in the sensing area of some certain type of sensor network, so that he can be alert to his privacy. However, he doesn't want a disclosure of his position as a pay of the judgement.

5.5.2 Who's Bob

The sensor network may have commercial or military privacy, so its specific sensing area can't be inferred by the originator. To give the originator a choice to test privacy state, the sensor network should have a central server to act as Bob in secure two-party point-inclusion problem. Besides, a trustful third party can also act as Bob simultaneously for multiple sensor networks. No matter which server is chosen, there should be a jural regulation that the sensor network must have a server when it is deployed.

Though Alice is safe because Bob can't learn her input information in the protocol, Bob may still have privacy concern for himself. During one single execution of the protocol, Alice can learn nothing about the specific area of Bob. However, if Alice continuously initiates the protocol and tests whether she is in the area, she may outline some parts of the area. As a result, the server, which is acting as Bob, should restrict the initiation times Alice can have.

5.5.3 The Architecture

Our architecture starts from the notion of providing a general scheme to test originator's privacy state in pervasive sensor networks. Fig 5.4 is a sketch map of the architecture. When one originator passes through the pervasively deployed sensor networks, he will need one single device (Personal Digital Assistant, for example) which can communicate with the servers. The servers may be a trustful third party for the sensor networks, or a central server of the network itself.

5.5.4 The Scheme

When one originator wants to test his privacy state, the following steps will be taken:

1) The PDA computes the current position of the originator by assistance of some location service and keeps it as its own privacy.

2) The PDA searches the central servers or the third party servers of the sensor networks. If it finds any, it initiates the protocol of the secure two-party point-inclusion problem.

3) The server checks whether the PDA has overused its initiation times. If not, the server agrees to continue the protocol.

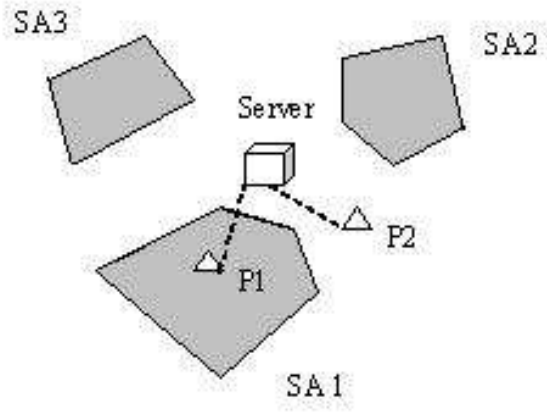

Figure 5.4: *The Architecture. Both originators, p_1 and p_2, have their own PDAs. PDA can communicate with the server.*

4) At the end of the protocol, the PDA concludes whether the originator is inside the sensing areas of some types of sensor networks. If he is inside, the PDA shows he is on the alert state of his privacy, otherwise, the PDA shows he is on the leisure state.

5) If the originator is on the alert state, the PDA also tells him a message about what type of his private information (voice, motion, body temperature, etc) is being observed. The PDA acquires this message from the server.

6) The originator decides whether to protect his privacy.

5.6 Evaluation of Our Scheme

Our Scheme assumes that the two parties, PDA and server, are semi-honest. Being semi-honest means the party follows the protocol properly with the exception that it keeps a record of all its intermediate computations and might try to derive other parties's private inputs from the record. At the end of the protocol, nothing more than the included-or-not information can be learned by the two parties. Initiation times of PDA are also checked by server, so sensing area privacy is further preserved. Thus, our scheme can ensure the confidentiality of the private inputs of PDA and server.

The company can choose whether to trust a third party server before the sensing area information is given to that server. If the company can't, it must have a central server for itself.

5.7 Chapter Summary and Future Work

In the near future sensor networks will be pervasively deployed to provide people the convenience of accessing information anytime and anywhere. People will have a dynamic concern about their own privacy, whereas it is another currency that sensors will become invisible or should be hidden due to the privacy of themselves. A general scheme is required to let people be aware of whether they should be alert on their private activities.

In this chapter, we discussed the privacy issues in pervasive sensor networks and introduced some terminologies such as originator privacy and sensing area privacy. Then we proposed a scheme to test the state of originator privacy in the case of sensing area privacy. Our scheme is general in that it is applicable for any types of sensor networks. In the assumption of two semi-honest parties, our scheme is also characteristic of confidentiality.

A large number of security-related problems are still open. Denial-of-service attack on the server may be employed by an adversary. The adversary may also trick the originator by fabricating a server, so authentication of the server should also be considered. The protocol solves only a point-inclusion problem on the planar surface. Some cubic solutions should also be studied.

Chapter 6

Conclusions and Future Work

In this chapter, we give a summary on our contributions in this thesis, and put forward some future directions on our work.

6.1 Contributions

This thesis contributes to formal definitions, new methods, efficient solutions, strengthened security for some specific privacy preserving computation problems in applications of e-bidding and databases. Related work lacks rigorous argument on the security, especially security under the malicious attacks. So for these problems we define their security under two models (semi-honest and malicious), employ cryptographic tools to construct more efficient solutions, and use cryptographic ways to formally prove the security. We are the first to achieve an efficient secure two-party vector dominance protocol in the malicious model, without employing the circuit evaluation techniques and causing unreasonable computation and communication costs. Our protocols for privacy preserving set intersection in the semi-honest and malicious model are more efficient than the related work. We propose an efficient solution for the problem of privacy preserving tuple matching, and are the first to address the problem of privacy preserving threshold attributes matching, and compared with solutions derived from related work, our proposed protocols for the two problems have less complexities and achieve reasonable responding time in the practical applications. We also employ our protocol for secure two-party vector dominance as a building block to guarantee the two contradictive privacy concerns (sensing area privacy and originator privacy) in wireless sensor network.

Specifically, our contributions can be listed in details as follows:

1) In the first part of work, we define the problem of secure two-party vector dominance (STVD) in terms of completeness, soundness and security in both semi-honest model and malicious model. we propose an STVD protocol which is proved to be overwhelmingly complete and sound, and be secure in the semi-honest model. We propose an STVD protocol which is proved to be overwhelmingly complete and sound, and be secure in the semi-honest model. Given K is the length of each element in the vector, in $K + 1$ parallel execution, our protocol has higher efficiency compared with a derived solution from [84] and another solution from [4]. We also fix our protocol to be secure against malicious behaviors in multi-commodity private bidding, and the fixed protocol is also proved to be overwhelmingly complete and

sound. In $K + 1$ parallel execution, our protocol has the same level of efficiency compared with the derived solution from [84].

2) In the second part of work, we give formal definitions of privacy preserving set intersection (PPSI) in both the semi-honest and malicious models. We propose efficient PPSI protocols for the two models respectively, and formally prove their security according to their definitions. In a quantitatively analyzing experiment, our PPSI protocol for the semi-honest model saves 81% and 63% computation costs, 17% and 20% communication costs in comparison with [62] and [36]. Our PPSI protocol for the malicious model keeps the same level of complexity as our PPSI protocol for the semi-honest model, that is, it has a computation cost of $O(cSNlg\mathcal{N})$ modular multiplications, and a communication cost of $O(cSNlg\mathcal{N})$ bits. c is the number of malicious parties, S is the size of dataset on each party, N is the total number of parties, and $lg\mathcal{N}$ is the length of modulus in the cryptosystem. In [63] the PPSI protocol for the malicious model has a computation cost of $O(cS^2lg\mathcal{N})$ modular multiplications, and also a communication cost of $O(cSNlg\mathcal{N})$ bits.

3) In the third part of work, our proposed protocol for the privacy preserving tuple matching (PPTM) problem has a lower computation cost than two related solutions in [62] and [63], and keeps the same security defined under Secure Multi-party Computation (SMC), without increasing intolerable communication cost. According to our experiments on a moderate-scale database, the dominant time cost is on the computation, rather than on the communication of our protocol. The computation of our PPTM protocol need $O(NS^2)$ modular exponentiations, but in [62] and [63] both of them need $O(N^2S^2)$ modular exponentiations. N is the total number of parties, and S is the number of tuples on each party. Our PPTM protocol has some increased communication bits, but they can be transferred within a few seconds.

   To our knowledge, we are the first to talk about the privacy preserving threshold attribute matching (PPTAM) problem, though a solution can be derived from the related techniques in [62]. Our proposed protocol for the PPTAM problem has lower computation and communication costs than the derived solution, while keeping the same level of SMC security. By our experiments, our PPTAM protocol has a reduction of about 71.3% in computation, and a reduction of about 75% in communication, in comparisons with the derived solution.

4) In the final part of work, we explore the issues of sensing area privacy and originator privacy in wireless sensor networks. The two issues conflict with each other, but are all critical concerns in practical applications. To our knowledge, we are the first to address them simultaneously. We provide a solution for the originator to test his privacy state in the case of sensing area privacy, and then judge whether he is infringed by his own initiatives. Our solution is based on the secure two-party point-inclusion problem, which employs our STVD protocol in the first part of work as a building block. To our knowledge, our solution is the first to guarantee the two contradictive privacy concerns without tampering with any of them.

6.2 Future Research Directions

In this thesis, we have proposed solutions to protect the participants' privacy in some basic computation problems. The common characteristics of these problems are that the inputs are distributed on different participants, the participants will claim privacy on their own inputs, but they are also in need of sharing some kind of information on these inputs. In practical applications, the information need to be shared may be as simple as the set intersection, or as complex as a mathematical model that can provide predictive decisions. Therefore, we will make for a more general research named as *Privacy Preserving Distributed Information Sharing* (PPDIS), and address the abundant sub-problems inside it.

From the research of this thesis, we notice that security and efficiency are two indispensable requirements for a solution to the privacy preserving computation problems. Any participant wants to complete the computation without worrying about the leaking of his privacy, and within a sustainable response time even when the inputs are large-scale databases. We also notice that both the security and efficiency are technique-related. The threshold homomorphic encryption schemes we used are robust against various powerful attacks, but they have limitations in that:

1) they can't be additive homomorphic and multiplicative homomorphic simultaneously, thus they have restricted utilities in computations composed of basic arithmetics.

2) they may need excessive modular exponentiations in large-scale inputs, and make the response time of the solution to be intolerable for the participants.

In some related work, data perturbation techniques (e.g., data swapping in [17, 31], additive and multiplicative distortion in [1, 38, 61, 43], probability transition matrix in [27, 28, 80], and k-Anonymity in [87], etc) are employed instead of cryptographic primitives, but they may be not secure against some powerful adversaries. Therefore, in our research of PPDIS, an important direction is to improve the protection techniques for the original inputs, without compromising the security and efficiency of the holistic solutions.

Specifically, we will expand our research on PPDIS as follows:

1) *Doing comparisons between data perturbation techniques and cryptographic techniques.* The inputs of each party may be composed of hundreds or thousands of records. When used for protecting these inputs, cryptographic primitives will have unreasonable response times, but data perturbation techniques may make an adversary's attacks easier. We will compare the cryptography and data perturbation techniques considering their execution times and security levels, then improve the security of data perturbation techniques utilizing the anti-attack techniques achieved in the study of cryptography.

2) *Proposing secure and efficient solutions for some basic PPDIS problems.* One example of such problems is privacy preserving tuple matching, for which we have proposed a solution for the semi-honest model. We will define some more rigorous security requirements for these problems than the related work, and use cryptographic tools to get solutions which are secure against the malicious attacks of some dishonest parties.

3) *Proposing secure solutions for some large-scale data mining tasks.* We will employ the protocols we have achieved in the basic problems such as PPSI, PPTM, PPTAM, to solve complex data mining tasks (e.g., privacy preserving classification, clustering, association rule mining), and improve the efficiency and security of the related work in these research fields.

Bibliography

[1] R. Agrawal, R. Srikant,"Privacy-preserving data mining," Proceedings of the ACM SIGMOD Conference on Management of Data, pages 439–450, 2000.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubrarnanian, and E. Cayirci, Wireless sensor networks: A survey, *Computer Networks*, Elsevier Science, 38(4), 2002, 393-422.

[3] G. Alia and E. Martinelli, "Fast modular exponentiation of large numbers with large exponents", in *Journal of Systems Architecture: the EUROMICRO Journal*, 47(14-15), pp. 1079 - 1088, 2002.

[4] M. J. Atallah and W. Du. "Secure Multi-party Computational Geometry". in *Proc. of the 7th International Workshop on Algorithms and Data Structures*, vol. 2125, pp. 165-179, LNCS, Springer-Verlag, 2001.

[5] A. R. Beresford and F. Stajano, "Location Privacy in Pervasive Computing", in *IEEE Pervasive Computing*, 2(1): 46-55, 2003.

[6] I. F. Blake and V. Kolesnikov. "Strong Conditional Oblivious Transfer and Computing on Intervals". in *Advances in Cryptology - ASIACRYPT '04*, vol. 3329, pp. 515-529, LNCS, Springer-Verlag, 2004.

[7] C. Blundo, A. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences", in *Crypto 92*, 1992.

[8] D. Boneh and M. Franklin, "Efficient Generation of Shared RSA Keys". in *Crypto97*, LNCS 1294, pages 425C439. Springer-Verlag, Berlin, 1997.

[9] F. Boudot, B. Schoenmakers and J. Traor'e, "A Fair and Efficient Solution to the Socialist Millionaires' Problem", in *Discrete Applied Mathematics*, 111(1-2), pp. 23-36, 2001.

[10] C. Cachin. "Efficient Private Bidding and Auctions with an Oblivious Third Party". in *Proc. of the 6th ACM Conference on Computer and Communications Security*, pp. 120-127, ACM Press, 1999.

[11] S. A. Camtepe and B. Yener, "Combinatorial design of key distribution mechanisms for wireless sensor networks", in *9th European Symposium on Research Computer Security*, 2004.

[12] R. Canetti. "Security and Composition of Multi-party Cryptographic Protocols". in *Journal of Cryptology*, 13(1): pp. 9-30, 2000.

[13] C. Castelluccia, E. Mykletun and G. Tsudik, "Efficient Aggregation of Encrypted Data in Wireless Sensor Networks", *ACM/IEEE Mobiquitous Conference*, July 2005, San Diego, USA.

[14] H. Chan, A. Perrig, D. Song, Random Key Predistribution Schemes for Sensor Networks, *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, 2003, 197.

[15] R. Cramer, I. Damgard, and J. Nielsen, "Multiparty Computation from Threshold Homomorphic Encryption", in *Advances in Cryptology - EUROCRYPT 2001*, LNCS, Springer, vol. 2045, pp. 280-300, 2001.

[16] R. Cramer and I. Damgard. "Zero-knowledge Proofs for Finite Field Arithmetic, or: can Zero-knowledge be for Free". in *Advances in Cryptology - CRYPTO '98*, vol. 1462, pp. 424-441, LNCS, Springer-Verlag, 1998.

[17] T. Dalenius and SP. Reiss, "Data-swapping: A Technique for Disclosure Control," Journal of. Statistical Planning and Inference, 6, pp. 73-85, 1982.

[18] A. D. Santis, G. D. Crescenzo, G. Persiano, M. Yung. "On Monotone Formula Closure of SZK". in *Proc. of the 35th Annual Symposium on Foundations of Computer Science*, pp. 454-465, IEEE Computer Society, 1994.

[19] A. Doan and A. Halevy, "Semantic Integration Research in the Database Community: A Brief Survey", in *AI Magazine, Special Issue on Semantic Integration*, 26(1), pp. 83-94, 2005.

[20] J. Domingo-Ferrer, "A provably secure additive and multiplicative privacy homomorphism", in *Information Security Conference*, LNCS 2433, pp.471-483, 2002.

[21] W. Du and M. Attalah, "Protocols for Secure Remote Database Access with Approximate Matching", in *Proc. of the 7th ACM CCS, the 1st Workshop on Security and Privacy in E-commerce*, 2000.

[22] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, A pairwise key pre-distribution scheme for wireless sensor networks. *Proceedings of the 10th ACM conference on Computer and communication security*, Washington D.C., USA, 2003, 42-51.

[23] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney, A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge, *IEEE INFOCOM'04*, Mar. 2004, Hongkong, China.

[24] W. Du, M. J. Atallah, Secure Multi-party Computation Problems and Their Applications: A Review and Open Problems. *Proceedings of the 2001 workshop on New security paradigms*, Cloudcroft, New Mexico, USA, 2001, 13-22.

[25] T. ElGamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", in *IEEE Transactions on Information Theory*, vol. IT-31, n. 4, pp. 469C472, 1985.

[26] L. Eschenauer, V. D. Gligor, A key-management scheme for distributed sensor networks, *Proceedings of the 9th ACM conference on Computer and communications security*, Washington, DC, USA, 2002, 41-47.

[27] A. Evfimievski, R. Srikant, R. Agrawal, J. Gehrke, "Privacy-Preserving Mining of Association Rules", in *8th ACM SIGKDD Int'l Conf. on Knowledge Discovery in Databases and Data Mining*, Edmonton, Canada, July 2002.

[28] A. Evfimievski, J. Gehrke, and R. Srikant, "Limiting privacy breaches in privacy preserving data mining", in *Proc. 22nd ACM Symposium on Principles of Database Systems (PODS 2003)*, pages 211–222.

[29] R. Fagin, M. Naor, and P. Winkler, "Comparing Information without Leaking It", in *Communications of the ACM*, 39(5): 77-85, 1996.

[30] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. Wright, "Secure Multiparty Computation of Approximations", in *Proc. of the 28th International Colloquium on Automata, Languages and Programming (ICALP 2001)*, pp. 927-938, 2001.

[31] S. E. Fienberg, J. McIntyre: "Data Swapping: Variations on a Theme by Dalenius and Reiss", Privacy in Statistical Databases 2004: 14-29.

[32] M. Fischlin. "A Cost-effective Pay-per-multiplication Comparison Method for Millionaires". in *Proceedings of the 2001 Conference on Topics in Cryptology: The Cryptographers Track at RSA*, vol. 2020, pp. 457-472, LNCS, Springer-Verlag, 2001.

[33] P. Fouque, G. Poupard and J. Stern, "Sharing Decryption in the Context of Voting or Lotteries", in *Proc. of the 4th International Conference on Financial Cryptography*, pp. 90 - 104, 2000.

[34] P. Fouque and D. Pointcheval, "Threshold Cryptosystems Secure against Chosen-ciphertext Attacks", in *Proc. of Asiacrypt 2001*, pp. 351 - 368, 2001.

[35] P. Fouque and J. Stern, "Fully Distributed Threshold RSA under Standard Assumptions", in *Asiacrypt2001*, LNCS, Springer-Verlag, Berlin, 2001.

[36] M. Freedman, K. Nissim and B. Pinkas, "Efficient Private Matching and Set Intersection", in *Proc. of Eurocrypt '04*, LNCS, Springer, vol. 3027, pp. 1 - 19, 2004.

[37] E. Fujisaki and T. Okamoto. "Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations". in *Advances in Cryptology - CRYPTO '97*, vol. 1294, pp. 16-30, LNCS, Springer-Verlag, 1997.

[38] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. in *The Third IEEE International Conference on Data Mining*, 2003.

[39] B. Lai, S. Kim, and I. Verbauwhede, "Scalable session key construction protocol for wireless sensor networks", In *IEEE Workshop on Large Scale RealTime and Embedded Systems (LARTES)*, Austin, Texas, December 2002.

[40] Y. Lindell. "Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation". in *Journal of Cryptology*, 16(3): pp. 143-184, 2003.

[41] D. Liu, and P. Ning, "Location-based pairwise key establishment for static sensor networks", in *1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003.

[42] D. Liu, and P. Ning, "Establishing pairwise keys in distributed sensor networks", in *10th ACM conference on Computer and communications security CCS03*, 2003.

[43] K. Liu, H. Kargupta, J. Ryan, Random projection-based multiplicative data perturbation for privacy preserving distributed data mining, IEEE Transactions on Knowledge and Data Engineering, 18(1):92C106, 2006.

[44] R. Gennaro, S. Jarecki, H. Krawczyk H and T. Rabin. "Secure Distributed Key Generation for Discrete-log Based Cryptosystems". in *Advances in Cryptology - EUROCRYPT'99*, vol. 1592, LNCS, pp. 295-310, 1999.

[45] J. Girao, D. Westhoff, and M. Schneider, "CDA: Concealed Data Aggregation for Reverse Multicast Traffic in Wireless Sensor Networks", *40th International Conference on Communications, IEEE ICC 2005*, May 2005, Korea.

[46] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen, "On Secure Scalar Product Computation for Privacy-Preserving Data Mining", in *Proc. of ICISC*, 2004.

[47] O. Goldreich, "Foundations of Cryptography: Volume 1, Basic Tools", Cambridge University Press, 2001.

[48] O. Goldreich. "Foundations of Cryptography: Volume 2". Cambridge University Press, 2001.

[49] O. Goldreich, S. Micali, and A. Wigderson, "How to Play Any Mental Game", in *Proc. of 19th STOC*, pp. 218-229, 1987.

[50] S. Goldwasser, "Multi-party Computations: Past and Present", in *Proc. of 16th annual ACM symposium on Principles of distributed computing*, pp. 1-6, 1997.

[51] M. Gruteser and D. Grunwald, Anonymous Usage of Location-based Services Through Spatial and Temporal Cloaking. *Proceedings of the First International Conference on Mobile Systems, Applications, and Services*, San Francisco, CA, USA, May 2003.

[52] T. Hartman and R. Raz, "On the Distribution of the Number of Roots of Polynomials and Explicit Weak Designs", in *Random Structures and Algorithms*, Vol. 23 (3), pp. 235 - 263, 2003.

[53] M. Hernandez and S. Stolfo, "The Merge/purge Problem for Large Databases", in *Proc. of the ACM SIGMOD International Conference on Management of Data*, pp. 127-138, 1995.

[54] S. Hohenberger and S. A. Weis, "Honest-Verifier Private Disjointness Testing without Random Oracles", in *Workshop on Privacy Enhancing Technologies (PET)*, 2006.

[55] M. H. Ibrahim. "Two-Party Private Vector Dominance: The All-Or-Nothing Deal", in *Proc. of the 3rd International Conference on Information Technology: New Generations*, pp. 166-171, IEEE Computer Society, Apr. 2006.

[56] P. Indyk and D. Woodruff, "Polylogarithmic Private Approximations and Efficient Matching", in *Proc. of the Third Theory of Cryptography Conference (TCC 2006)*, LNCS, Springer, vol. 3876, pp. 245-264, 2006.

[57] I. Ioannidis and A. Grama. "An Efficient Protocol for Yao's Millionaires Problem". in *Proc. of the 36th Hawaii Internatinal Conference on System Sciences*, vol. 07, no. 7, pp. 205a, 2003.

[58] M. Jakobsson, A. Juels, "Mix and Match: Secure Function Evaluation via Ciphertexts", in *Advances in Cryptology - ASIACRYPT 2000*, vol. 1976, pp. 162-177, LNCS, Springer-Verlag, 2000.

[59] J. M. Kahn, R. H. Katz, and K. S. J. Pister, Next century challenges: Mobile Networking for "Smart Dust", *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking(MobiCom)*, Seattle, United States, 1999, 271-278.

[60] A. Kiayias and A. Mitrofanova, "Testing disjointness of private datasets", in *Proc. of Financial Cryptography (FC 2005)*, LNCS, Springer, vol. 3570, pp. 109C124, 2005.

[61] J. J. Kim and W. E. Winkler, "Multiplicative Noise for Masking Continuous Data", in *Statistical Research Division*, US Bureau of the Census, Washington DC 20233.

[62] L. Kissner and D. Song, "Privacy-Preserving Set Operations", in *Advances in Cryptology - CRYPTO 2005*, LNCS, Springer, vol.3621, pp. 241-257, 2005.

[63] L. Kissner and D. Song, "Privacy-Preserving Set Operations", in *Technical Report CMU-CS-05-113*, Carnegie Mellon University, June 2005.

[64] K. Kurosawa, W. Ogata, "Bit-Slice Auction Circuit", in *Proc. of European Symposium on Research in Computer Security (ESORICS 2002)*, vol. 2502, pp. 24-38, LNCS, Springer-Verlag, 2002.

[65] H. Y. Lin and W. G. Tzeng. "An Efficient Solution to The Millionaires' Problem Based on Homomorphic Encryption". in *Proc. of Applied Cryptography and Network Security 2005*, vol. 3531, pp. 456-466, LNCS, Springer-Verlag, 2005.

[66] Y. Lindell. "Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation". in *Journal of Cryptology*, 16(3): pp. 143-184, 2003.

[67] H. Lipmaa, "Verifiable Homomorphic Oblivious Transfer and private Equality Test", in *Advances in Cryptography ASIACRYPT 2003*, pp. 416-433, 2003.

[68] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks", *Proceedings of the 10th ACM conference on Computer and communication security*, Washington D.C., USA, Oct. 2003, 52-61.

[69] W. Lou, W. Liu, Y. Fang, "SPREAD: Enhancing Data Confidentiality in Mobile Ad Hoc Networks", *IEEE INFOCOM 2004*, 2004.

[70] A. Mahimkar, T. S. Rappaport, "SecureDAV: A Secure Data Aggregation and Verification Protocol for Sensor Networks", *Proceedings of IEEE Global Telecommunications Conference (Globecom) 2004*, Nov, 2004, Dallas, TX, USA.

[71] A. Menezes, P. van Oorschot, and S. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996.

[72] R. P. Minch, Privacy Issues in Location-Aware Mobile Devices, *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences(HICSS'04)*,Track 5, Volume 5, Jan. 2004.

[73] E. Mykletun, J. Girao, and D. Westhoff, "Public key based cryptoschemes for data concealment in wireless sensor networks", in *IEEE International Conference on Communications (ICC2006)*, June 2006, Turkey.

[74] M. Naor and B. Pinkas, "Oblivious Transfer and Polynomial Evaluation", in *Proc. of the 31st Annual ACM Symposium on Theory of Computing*, pp. 245-254, 1999.

[75] P. Paillier, "Public-key Cryptosystems based on Composite Degree Residuosity Classes", in *Proc. of Asiacrypt 2000*, pp. 573-584, 2000.

[76] K. Peng, C. Boyd, E. Dawson and B. Lee. "An Efficient and Verifiable Solution to the Millionaire Problem". in *Proc. of the 7th International Conference on Information Security and Cryptology*, vol. 3506, pp. 51-66, LNCS, Springer-Verlag, 2004.

[77] A. Perrig, J. Stankovic, and D. Wagner, Security in wireless sensor networks, *Communications of the ACM*, 47(6), Jun. 2004, 53-57.

[78] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, The Cricket location-support system, *Proceedings of the 6th annual international conference on Mobile computing and networking*, Boston, Massachusetts, USA, 2000, 32-43.

[79] D. Randall, "Efficient Generation of Random Nonsingular Matrices", in *Random Structures and Algorithms*, vol. 4(1), pp. 111-118, 1993.

[80] S. Rizvi and J. Haritsa,"Maintaining Data Privacy in Association Rule Mining", in *Proc. of 28th Intl. Conf. on Very Large Databases (VLDB)*, August 2002.

[81] Y. Sang, H. Shen, Z. Zhang. "An Efficient Protocol for the Problem of Secure Two-party Vector Dominance". in *Proc. of 6th International Conference on Parallel and Distributed Computing Applications and Technologies*, pp. 488-492, IEEE Computer Society, Dec. 2005.

[82] S. Sarawagi and A. Bhamidipaty, "Interactive deduplication using active learning", in *Proc. of the Eighth ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pp. 269-278, 2002.

[83] C. P. Schnorr. "Efficient Signature Generation by Smart Cards". in *Journal of Cryptology*, 4(3): pp. 161-174, 1997.

[84] B. Schoenmakers and P. Tuyls. "Practical Two-Party Computation based on the Conditional Gate". in *Advances in Cryptology - ASIACRYPT 2004*, vol. 3329, pp. 119-136, LNCS, Springer-Verlag, 2004.

[85] A. Shamir, "How to Share a Secret", in *Communications of the ACM*, 22:612C613, November 1979.

[86] V. Shoup, "Practical Threshold Signatures", in *Eurocrypt2000*, LNCS 1807, pages 207C220. Springer-Verlag, Berlin, 2000.

[87] L. Sweeney. "k-anonymity: a model for protecting privacy", in *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10 (5), pp. 557-570, 2002.

[88] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, Lessons from a Sensor Network Expedition, *Proceedings of the First European Workshop on Sensor Networks(EWSN '04)*, Berlin, Germany, Jan. 2004.

[89] S. Tilak, N. Abu-Ghazaleh, and W. Heinzelman, A Taxonomy of Wireless Microsensor Network Models, *ACM Mobile Computing and Communications Review (MC2R 2002)*, 6(2), Apr. 2002, 28-36.

[90] C. D. Walter, "Exponentiation Using Division Chains", in *IEEE Transactions on Computers*, 47(7), pp. 757-765, 1998.

[91] M. Weiser, The Computer for the 21st Century, *Scientific American*, Sept.1991, 265(3), 94-104.

[92] A.C. Yao, "Protocols for Secure Computations", in *Proc. of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 160 - 164, 1982.

Publications

[1] Yingpeng Sang, Hong Shen, Laurence T. Yang, Yasuo Tan, Naixue Xiong: *An Efficient and Secure Protocol for Privacy Preserving Set Intersection*, accepted by Journal of Autonomic and Trusted Computing, American Scientific Publishers.

[2] Yingpeng Sang, Hong Shen, Yasuo Tan, Zonghua Zhang: *A Secure and Efficient Two-party Vector Dominance Protocol for Multi-commodity Private Bidding*, submitted to Computer Communications.

[3] Yingpeng Sang, Hong Shen, Yasuo Tan: *Privacy Preserving Tuple Matching in Distributed Database*, submitted to IEICE Transactions on Information and Systems.

[4] Yingpeng Sang, Hong Shen, Yasuo Tan, Naixue Xiong: *Efficient Protocols for Privacy Preserving Matching against Distributed Datasets*, Proc. of the Eighth International Conference on Information and Communications Security (ICICS'06), LNCS 4307, pp. 210-227, Raleigh, NC, USA, December 2006.

[5] Yingpeng Sang, Hong Shen, Yasushi Inoguchi, Yasuo Tan, Naixue Xiong, *Secure Data Aggregation in Wireless Sensor Networks: A Survey*. Proc. of the 7th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2006), IEEE CS, Taipei, Taiwan, December 2006.

[6] Yingpeng Sang, Hong Shen, Zonghua Zhang, *An Efficient Protocol for the Problem of Secure Two-party Vector Dominance*, Proc. of the 6th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2005), IEEE CS, pp. 488-492, Dalian, China, December 2005.

[7] Yingpeng Sang, Hong Shen, *A Scheme for Testing Privacy State in Pervasive Sensor Networks*, Proc. of the 19th International Conference on Advanced Information Networking and Applications (AINA 2005), IEEE CS, pp. 644-648, Taipei, Taiwan, March 2005.

[8] Yingpeng Sang, Hong Shen, Pingzhi Fan, *Novel Impostors Detection in Keystroke Dynamics by Support Vector Machine*, Proc. of the 5th International Conference on Parallel and Distributed Computing: Applications and Technologies (PDCAT 2004), LNCS 3320, pp. 666-669, Singapore, December 2004.