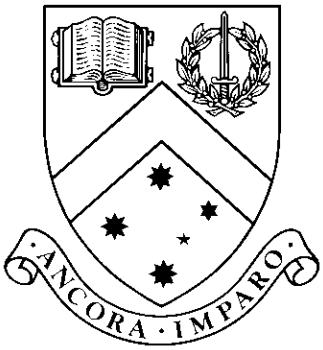
DAG: A General Model for Privacy-Preserving Data Mining

by

Sin Gee Teo, M.Eng.(Software)



Thesis

Submitted by Sin Gee Teo

for fulfillment of the Requirements for the Degree of

Doctor of Philosophy (0190)

Supervisor: Dr. Vincent Cheng-Siong Lee

Associate Supervisor: Dr. Jianneng Cao

Clayton School of Information Technology
Monash University

May, 2016

For God, the almighty and my family members.

Contents

List of Tables

List of Figures

Abstract

"Research consists in seeing what everyone else has seen, but thinking what no one else has thought." **—Albert Szent-Gyorgyi**

Rapid advances in automated data collection tools and data storage technology has led to the wide availability of huge amount of distributed data owned by different parties. Data mining can use the distributed data to discover rules, patterns or knowledge that are normally not discovered data owned by a single party. Thus, data mining on the distributed data can lead to new insights and economic advantages. However, in recent years, privacy laws have been enacted to protect any individual sensitive information from privacy violation and misuse. To address the issue, many have proposed privacy-preserving data mining (PPDM) based on secure multi-party computation (SMC) that can mine the distributed data with privacy preservation (i.e., privacy protection). However, most SMC-based solutions are ad-hoc. They are proposed for specific applications, and thus cannot be applied to other applications directly. Another limitation of current PPDM is with only a limited set of operators such as $+, -, \times$ and log (logarithm). In data mining primitives, some functions can involve operators such as / and $\sqrt{}$. The above issues have motivated us to investigate a general SMC-based solution to solve the current limitations of PPDM.

In this thesis, we propose a general model for privacy-preserving data mining, namely as DAG. We apply a hybrid model that combines the homomorphic encryption protocol and the circuit approach in DAG model. The hybrid model has been proven efficient in computation and effective in protecting data privacy via the theoretical and experimental proofs. Specifically, our proposed research objectives are as follows:

(i) We want to propose a general model of privacy-preserving data mining (i.e., DAG) that consists of a set of secure operators. The secure operators can support many mining primitives. The two-party model which is the efficient and effective model is applied to develop secure protocols in DAG. Our secure operators can provide a

complete privacy under the semi-honest model. Moreover, the secure operators are efficient in computation.

(ii) We will integrate DAG model into various classification problems by proposing new privacy-preserving classification algorithms.

(iii) To make our DAG model that can support wider applications, we will integrate DAG into other application domains. We will integrate DAG into ant colony optimization (ACO) to solve the traveling salesman problem (TSP) by proposing a new privacy-preserving traveling salesman problem (PPTSP).

In this thesis, we present most results of the objectives mentioned above. The DAG model is general – its operators, if pipelined together, can implement various functions. It is also extendable – new secure operators can be defined to expand the functions the model supports. All secure operators of DAG are strictly proven secure via simulation paradigm (Goldreich, 2004). In addition, the error bounds and the complexities of the secure operators are derived so as to investigate accuracy and computation performance of our DAG model. We apply our DAG model into various application domains. We first apply DAG into data mining classification algorithms such as support vector machine, kernel regression, and Naïve Bayes. Experiment results show that DAG generates outputs that are almost the same as those by non-private setting, where multiple parties simply disclose their data. DAG is also efficient in computation of data mining tasks. For example, in kernel regression, when training data size is 683,093, one prediction in non-private setting takes 5.93 sec, and that by our DAG model takes 12.38 sec. In the experiment of PPTSP, a salesman can find the approximate optimal traveled distance without disclosing any city locations in TSP. Various domain applications studies show that our DAG is the general model yet efficient for secure multi-party computation.

DAG: A General Model for Privacy-Preserving Data Mining

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Sin Gee Teo
May 27, 2016

Acknowledgments

Pursuing a PhD degree has been given me an incredible journey to discover and find the solutions for problems. Many research skills have been developed during my PhD years. I would like to thank many people and organizations who helped me to finish my research degree possible.

Monash University, Australia and **Institute for Infocomm Research, Singapore** offered me a scholarship to pursue my higher degree. I had spent two years to do my research in Monash University, Australia. Another two years, I attached to Institute for Infocomm Research, Singapore

I would like to express my deep gratitude and appreciation to my supervisors, **Associate Professor Vincent Cheng-Siong Lee** and **Dr. Jianneng Cao**. They were great mentors to me. They also always guided me to improve my research skills especially in the areas of the paper writing and communication skills which I could convey my ideas clearly and precisely to my readers. Dr. Cao always patiently reviewed and commented my conference papers and verified the ideas and the proofs I developed. I was inspired of our valuable weekly discussions that could lead to discover good solutions of my research problems. Besides the research, I appreciated that my supervisors gave me good advices on my future career path. All valuable lessons that I learned from my supervisors will grow me as a good research scientist in the future.

I would also like to express my special thanks to **Dr. Shuguo Han** who was a research scientist in Institute for Infocomm Research, Singapore. Dr. Han is an expert in the field of secure multi-party computation. He always shared the ideas in solving the difficult problems of secure multi-party computation. Our discussions had helped me to resolve many of my research problems in the later stage of my PhD years.

Dr. Shukai Li and **Dr. Xiaoli Li** are research scientists in Institute for Infocomm Research, Singapore. They are experts in the field of data mining. They always gave good advices in solving some of my research problems. Besides, Dr. Xiaoli Li also helped to review my papers before submitting to the conferences. I would like to express a special appreciation to **Dr. Brian Jenney**. He is willing to spend his valuable time to do a significant amount of work in proof-reading on my thesis.

Last but not least, I would like to express thanks to **Kaiyun Teh**, **Mathew & Gini Thomas**, **Chuenyong Liong**, **Dr. Tony Luo**, **Mark Yu**, **Camilla Chen**, **Wei Qiu**, **Joyce Qiu**, **Igor & Lydia Wilski**, **my grandma**, **my parent** and many more who always supported and encouraged me during my PhD years. I thanked also to my God (Yahweh) who always gave wisdom, good health and tenacity to me that I could complete my PhD degree according to the plan.

Thank you all.

Chapter 1

Introduction

Section 1.1 presents the motivation of a general model for privacy-preserving data mining. We discuss the research objectives and contributions of this thesis in Section 1.2. The list of our publications and technical reports is detailed in Section 1.3. In the last section we present our thesis organization.

1.1 Motivation

In many real-world applications, data are distributed across multiple parties. These parties have a strong willingness of sharing their data, so as to have a global view of the data and discover knowledge that cannot be mined from the data of any single party. Distributed data mining on multi-party datasets can lead to new insights and economic advantages. For instance, the homeland security (Seifert, 2007) can possibly discover potential terrorists via the distributed data mining. Another example is that banks can detect money laundering activities (Vaidya and Clifton, 2004) by which illegal transaction patterns are discovered by the distributed data mining. The distributed data mining is widely applied in various domains such as in bioinformatics (Hsu, 2006), risk management (Rud, 2001), business intelligence (Shmueli et al., 2006), discovery-based science and many more.

However, many parties are not willing to share data that may contain sensitive information; e.g., sharing patient medical records may affect a patient insurance coverage or employment, and sharing financial transactions of a person may cause him to become a victim to fraud or identify theft. Thus, directly sharing them may violate personal privacy (Aggarwal and Yu, 2008; Clifton et al., 2002). Government agencies have enacted laws to protect personal privacy. Well known representatives are HIPAA (Congress, 1996) of the Unites States and ECHR (ECHR, 2014) of the Europen Union. Therefore, data sharing for analytics needs to be carried out in a privacy-preserving way. However, there is no

100% guarantee that all semi-honest parties will comply to the privacy laws at all times. We still need techniques to enforce the laws.

Randomization (Agrawal and Srikant, 2000) and secure multi-party computation (SMC) (Goldreich, 2004) are two common techniques that can enforce the privacy laws. In the randomization, noises are added to distort the original data in achieving security but lose accuracy. This technique obviously can reduce the data utility rate especially in affecting data mining result accuracy. SMC is a fundamental field in cryptography. It allows multiple parties to jointly compute a function over their inputs, while keeping respective inputs of the parties private. Such a property is very effective in protecting personal privacy. Thus, SMC can yield higher accuracy at the expense of more communication and computation costs. SMC has been extensively applied in many data mining algorithms to protect data privacy, such as decision tree (Vaidya, Clifton, Kantarcioglu and Patterson, 2008), Naïve Bayes (Vaidya, Kantarcioglu and Clifton, 2008), support vector machine (Teo et al., 2013), and association rule mining (Kantarcioglu and Kardes, 2006).

However, many proposed SMC solutions are ad-hoc and specific to tasks. They cannot be directly applied to other tasks; e.g., the SMC protocol in privacy-preserving decision tree (Vaidya, Clifton, Kantarcioglu and Patterson, 2008) is not applicable in a privacy-preserving support vector machine (Teo et al., 2013). Developing a new SMC protocol together with the protocol analysis is a time-consuming task. Thus, any reusable SMC protocol is important that can directly reduce time spent on the protocol. Another limitation of the current SMC protocols of the privacy-preserving data mining (PPDM) algorithms is provided only by a limited set of operators such as $+, \times$, and log (logarithm) (Section 3.1). These existing operators are difficult to pipeline together to perform more complex functions, e.g., functions involving $\times$ and log. These operators are also unable to support some functions in some data mining tasks, e.g., functions involve $\sqrt{}$ and /. Hence, the limitations restrict their scalability. In this thesis, we focus to investigate new secure operators of SMC that can compute more functions in data mining primitives.

The two-party protocol is an efficient and practical model (Pinkas et al., 2009) for secure multi-party computation (SMC). The circuit approach and the homomorphic encryption protocol are two common methods used in SMC (Section 2.5). The circuit approach (Sections 2.5.1 and 2.5.2) can support only limited functions in data mining primitives. Thus, in this thesis, we use a hybrid model to combine semi-homomorphic encryption

(Section 2.1) and the circuit approach to propose efficient and effective secure protocols of SMC. We use the two-party protocol (e.g., Alice and Bob) to propose a general model that consists of a set of secure operators for privacy-preserving data mining, namely as DAG (Chapter 4). The protocol analysis of each secure operator is also discussed in detail that includes its error analysis and complexity analysis. We apply DAG into various application domains. We propose privacy-preserving classification algorithms to solve different classification problems with privacy preservation by applying DAG (Chapter 5). Lastly, we also propose to visit the privacy-preserving traveling salesman problem (PPTSP) by DAG to securely find an optimal approximate traveled distance by a salesman in the traveling salesman problem (TSP) without disclosing any city locations (Chapter 6).

1.2 Research Objectives and Contributions

The goal of this thesis is to propose a DAG which is a general model for privacy-preserving data ming over distributed data. A set of secure operators of DAG based on secure multiparty computation (SMC) will be defined and added to DAG based on the security requirement of a secure operator (Definition 4.1). Our secure operators can be pipelined together to perform more complex functions in data mining primitives. In addition, a new operator derived based on Definition 4.1 can be added into the DAG model to support more functions in the future. We need to prove secure operators that are efficient in computation and effective in protecting data privacy. We will strictly prove the security of secure operators of the DAG model based on the semi-honest model via simulation paradigm (Goldreich, 2004). Furthermore, the error bounds and the complexities of the secure operators will be derived to investigate accuracy and computation performance of our DAG model. The effective and efficient DAG can determine whether it is applicable to many time-sensitive applications. Various domain applications integrated with DAG will be examined carefully. We will first integrate DAG with different classification algorithms. Subsequently, we will integrate DAG in other application domain. Our DAG model is general and extendable yet efficient, and potentially can be applied in wider applications. Specifically, in this thesis:

(i) In Chapter 4, we propose a DAG which a general model for privacy-preserving data mining. Our DAG consists of 3 types of nodes: source, sink, and operator. Source nodes are private inputs of the parties involved in the tasks. Sink nodes are the

outputs of the model. Operators are to provide functions. The nodes in the model are connected by edges, such that outputs of the upper stream nodes are the inputs of the downstream nodes. To keep the respective input of each party confidential, security requirements are enforced on the operators. Specifically, an operator is formulated as an SMC protocol (Yao, 1986), such that given the private inputs of multiple parties, the operator allows the involved parties to learn the operator output while keeping their respective inputs confidential.

(ii) We propose a set of operators of the DAG model based on the two-party protocol of SMC: secure addition, secure minus, secure multiplication, secure division, secure log (logarithm), secure power, secure bit-length, secure max and secure max location. We use a hybrid model that combines semi-homomorphic encryption and the circuit approach to develop efficient and effective secure operators of DAG. We use the efficient semi-homomorphic encryption scheme which is proven secure for our secure operators. The secure bit-length protocol, the secure max protocol and the secure max location protocol all require value comparison. Therefore, we apply the efficient integer comparison circuit (CMP) to securely compare values for the protocols. The secure bit-length is a sub-protocol for the secure division protocol, the secure log protocol, and also the secure power protocol.

(iii) In the DAG model, each secure operator can perform the function as specific. The secure operators can be pipelined together to perform more complex functions. Our secure operators can support integer values and non-integer values. Many homomorphic encryption schemes support only integer values. Therefore, we need to convert a floating value (non-integer value) into an integer value. To preserve the precision of operator output, we set necessary parameters and apply various techniques, like Taylor's series, on input of floating values. The error bound of every single operator, and also the error bound of the concatenation of the operators are discussed in detail. It turns out when two operators are concatenated, their accumulated error is bounded by the linear combination of their respective errors.

(iv) Our secure operators of DAG are strictly proven secure via simulation paradigm. We also use other security measurements such as computational indistinguishability and statistical indistinguishability; e.g., the outputs of secure operators are proven secure

using the statistical distance. Thus, our secure operators can provide a complete privacy under the semi-honest model. Moreover, secure operators of DAG are also evaluated in the experiment. Theoretical and experimental proofs show that our DAG is efficient in computation and effective in protecting data privacy.

(v) In Chapter 5, we integrate our DAG model into various data mining classification algorithms: support vector machine (SVM), kernel regression and Naïve Bayes. In SVM, we assume that data are partitioned arbitrary between two parties (i.e., Alice and Bob). To construct the global SVM model without disclosing any data of Alice and that of Bob, we apply secure multiplication of DAG to compute the Gram matrix. We apply secure division of DAG to securely compute the estimated function in the kernel regression. All training data are split horizontally between Alice and Bob in the kernel regression. We also apply secure operators of DAG model to build a classifier model and then use the model to test instances in Naïve Bayes. Again, all training data are split horizontally between Alice and Bob in Naïve Bayes. The secure operators applying in Naïve Bayes include secure multiplication, secure division, secure power, and other operators. The security analysis and complexity analysis of each classification algorithm integrated with DAG are discussed in detail. We evaluate privacy-preserving classification algorithms with different datasets. Experiments show that privacy-preserving classification algorithms by DAG can obtain almost the similar mining results by classification algorithms without privacy preservation. In addition, the DAG model is efficient in computation and effective to protect data privacy in the data mining tasks. Thus, our DAG model is a general model yet efficient for privacy-preserving data mining.

(vi) In Chapter 6, we also apply the DAG model into other application area. We integrate DAG into ant colony optimization (ACO) that uses a probabilistic method (evolution computation) to solve traveling salesman problem (TSP). To the best of our knowledge, we are first to apply privacy preservation into evolution computation. In TSP, a salesman requires to find the optimal distance to travel all in a given set of cities. We assume that cities contain locations which are sensitive. The cities are split between Alice and Bob. ACO is integrated with DAG that can allow the salesman to find the approximate optimal distance in visiting all cities without disclosing the

locations of Alice and that of Bob. The secure operators applying in ACO include secure multiplication, secure division, secure power, and other operators. The security analysis and complexity analysis of ACO integrated with DAG are discussed in detail. Lastly, we evaluate privacy-preserving traveling salesman problem (PPTSP) with different datasets. Experiments show that PPTSP by DAG can obtain almost the similar optimal traveled distances by ACO without privacy preservation. Thus, our DAG model can achieve high accuracy and be effective to protect data privacy in various application domains.

1.3 List of Publications and Technical Reports

In this thesis, our work is contributed to 1 journal (under review), 3 conference papers and 4 technical reports as follows.

1. **Sin G. Teo**, Jianneng Cao, Vincent C.S. Lee, "DAG: A General Model for Privacy-Preserving Data Mining", **submitted** to the Journal of IEEE Transactions on Knowledge and Data Engineering (**TKDE**) (under review).

2. **Sin G. Teo**, Jianneng Cao, Vincent C.S. Lee, "DAG: A Model for Privacy Preserving Computation", In Proceedings of the 22nd IEEE International Conference on Web Services (**ICWS'15**), 289-296, New York, USA, June 27-July 2, 2015.

3. **Sin G. Teo**, Shuguo Han, Vincent C.S. Lee, "Privacy Preserving Support Vector Machine Using Non-linear Kernels on Hadoop Mahout", In Proceedings of the 16th IEEE International Conference on Computational Science and Engineering (**CSE'13**), 941-948, Sydney, Australia, 3-5 December, 2013.

4. **Sin G. Teo**, Vincent C.S. Lee, Shuguo Han, "A Study of Efficiency and Accuracy of Secure Multiparty Protocol in Privacy-Preserving Data Mining", In Proceedings of the 26th IEEE International Conference on Advanced Information Networking and Applications Workshops (**WAINA'12**), 85-90, Fukuoka, Japan, 26-29 March, 2012.

5. **Sin G. Teo** Jianneng Cao, Vincent C.S. Lee, "Secure Log Operator", Joint Technical Report between Monash University, Australia and Institute for Infocomm Research, Singapore.

6. **Sin G. Teo** Jianneng Cao, Vincent C.S. Lee, "Privacy-Preserving Ant Colony in Distributed Setting", Joint Technical Report between Monash University, Australia and Institute for Infocomm Research, Singapore.

7. **Sin G. Teo** Jianneng Cao, Vincent C.S. Lee, "Privacy-Preserving Multi-party AntMiner", Joint Technical Report between Monash University, Australia and Institute for Infocomm Research, Singapore.

8. **Sin G. Teo** Jianneng Cao, Vincent C.S. Lee, "Improving Data Applicability with Privacy Preserving SVM using Parallelism and Concurrency", Joint Technical Report between Monash University, Australia and Institute for Infocomm Research, Singapore.

1.4 Thesis Organization

This thesis is organized as follows: In Chapter 2, the background knowledge of homomorphic cryptosystems, secure multi-party computation (SMC), adversary models of SMC and related work are presented. Chapter 3 presents a general survey of privacy-preserving data mining with secure building blocks. We propose a DAG which is a general model that consists of a set of operators for privacy-preserving data mining in Chapter 4. In Chapter 5, we propose privacy-preserving classification algorithms by DAG to solve different classification problems with privacy preservation. Ant Colony Optimization integrated with DAG to solve the traveling salesman problem (TSP) with privacy preservation is presented in Chapter 6. Lastly, Chapter 7 concludes the thesis and proposes some future work of our DAG model.

Chapter 2

Background

Homomorphic encryption allows specific types of operation to perform on ciphertexts (i.e., encrypted messages). The encrypted result of the operations, when decrypted, is equivalent to the result of operations performed on the messages (i.e., plaintexts). Given an encryption scheme (M, C, K, E, D) where M is the plaintext space, C is the ciphertext space, K is the key space, and E and D are encryption and decryption algorithms respectively, the plaintext M and the ciphertext C form groups $(M, \circ)$ and $(C, \diamond)$ respectively. The encryption algorithm E then can apply to M by mapping the group $(M, \circ)$ to the group $(C, \diamond)$, such that $E_k : M \mapsto C$ where $k \in K$ can be either a public key (e.g., in a public-key cryptosystem) or a secret key (e.g., in a secret-key cryptosystem). Thus, the encryption scheme is homomorphic by

$$E_k(m_1) \circ E_k(m_2) = E_k(m_1 \diamond m_2), \tag{2.1}$$

where m_1 and m_2 are plaintexts in M and $k \in K$. Currently two types of the homomorphic encryption have been proposed, semi-homomorphic encryption (SHE) and fully-homomorphic encryption (FHE).

In this chapter, we first survey semi-homomorphic encryption (SHE) and fully-homomorphic encryption (FHE) in Sections 2.1 and 2.2, respectively. We discuss oblivious transfer in Section 2.3. Secure multi-party computation (SMC) is discussed in Section 2.4. We discuss three SMC models and two adversary models of SMC in Sections 2.5 and 2.6, respectively. Lastly, we conclude this chapter with discussion in Section 2.7.

2.1 Semi-Homomorphic Encryption (SHE)

In semi-homomorphic encryption, it basically consists of three functions; key generation, encryption and decryption. This section discusses four common schemes in the semi-homomorphic encryption: Paillier Encryption Scheme (Section 2.1.1), Goldwasser-Micali encryption scheme (Section 2.1.2), ElGamal encryption scheme (Section 2.1.3), and Boneh-Goh-Nissim Encryption Scheme (Section 2.1.4).

2.1.1 Paillier Encryption Scheme

The Paillier encryption scheme (Paillier, 1999) is a probabilistic public-key encryption based on the decisional composite residuosity assumption. The assumption states that finding n-th residue classes is computationally intractable.

Key Generation. In the scheme, key generation can divide into a few steps. Initially two large prime numbers p and q are randomly and independently selected, such that,

$$\gcd(p \cdot q, (p-1) \cdot (q-1)) = 1, \tag{2.2}$$

where both p and q are equal length and $p \neq q$. Next step, we compute

$$n = p \cdot q, \lambda = \text{lcm}(p-1, q-1), \tag{2.3}$$

where lcm is the least common multiple. To allow n that can divide the order of $g \in \mathbb{Z}^*_{n^2}$, the modular multiplicative inverse can be used to check,

$$\upsilon = (L(g^{\lambda} (\text{mod } n^2)))^{-1} (\text{mod } n), \tag{2.4}$$

where L is the function (i.e., $L(u) = \frac{u-1}{n}$). The output of L is the quotient of $\frac{u-1}{n}$. The public and private keys are (n, g) and (λ, υ), respectively; e.g., $g = n + 1$, $\lambda = \varphi(n) = (p-1) \cdot (q-q)$, and $\upsilon = \varphi(n)^{-1} \bmod n$.

Encryption. Given the public key (g, n), the plaintext message m can be encrypted to

$$c = g^m \cdot r^n (\text{mod } n^2), \tag{2.5}$$

where $m \in \mathbb{Z}_m$ and $r \in \mathbb{Z}_n^*$ is a random number.

Decryption. The ciphertext c can be decrypted to the message m as follows,

$$m = L(c^{\lambda} (\text{mod } n^2)) \cdot \upsilon (\text{mod } n), \tag{2.6}$$

where the ciphertext $c \in \mathbb{Z}_{n^2}^*$. In binomial theorem, we can get $y = (1+n)^x = 1 + nx \ (\text{mod } n^2)$ and then x is computed as

$$x = \frac{y-1}{n} \ (\text{mod } n), \tag{2.7}$$

where $x \in \mathbb{Z}_n$. Thus, the function $L((1+n)^x (\text{mod } n^2)) = x \ (\text{mod } n)$. The ciphertext c can be decrypted to the plaintext message m,

$$\begin{aligned} L(c^{\lambda} (\text{mod } n^2)) \cdot \upsilon &= L((g^m r^n)^{\lambda} (\text{mod } n^2)) \cdot \lambda^{-1} \\ &= L((g^{m\lambda}) (\text{mod } n^2)) \cdot \lambda^{-1} \\ &= \lambda \cdot m \cdot \lambda^{-1} = m \ (\text{mod } n), \end{aligned} \tag{2.8}$$

where $g = n + 1$.

Homomorphic Property. Let m_1 and m_2 be two messages and (pk, sk) (i.e., (public key, secret key)) be a pair of keys generated by two large prime numbers p and q in Paillier encryption scheme. We use $E[.]$ and $D[.]$ to denote encryption and decryption functions respectively, and set $n = pq$. The messages m_1 and m_2 can encrypt to $E[m_1, pk] = g^{m_1} r_1^n (\text{mod } n)$ and $E[m_2, pk] = g^{m_2} r_2^n (\text{mod } n)$, respectively, where r_1^n and r_2^n are randomly selected from $\mathbb{Z}_n^*$.

- **Homomorphic Addition:** The product of two ciphertexts of m_1 and m_2 decrypts to the sum of m_1 and m_2 (i.e. $m_1 + m_2$), such that

$$D[sk, E[m_1, pk] \cdot E[m_2, pk] \ (\text{mod } n^2)] = m_1 + m_2 \ (\text{mod } n), \tag{2.9}$$

where

$$\begin{aligned} E[m_1, pk] \cdot E[m_2, pk] &= (g^{m_1} r_1^n)(g^{m_2} r_2^n) (\text{mod } n^2) \\ &= (g^{m_1+m_2}(r_1 r_2)^n) (\text{mod } n^2) \\ &= E[m_1 + m_2, pk]. \end{aligned} \tag{2.10}$$

- **Homomorphic Multiplication:** The ciphertext of m_1 raised to the power of m_2 decrypts to the product of m_1 and m_2 (i.e. $m_1 \cdot m_2$), such that

$$D[sk, E[m_1, pk]^{m_2} \ (\text{mod } n^2)] = m_1 \cdot m_2 \ (\text{mod } n), \tag{2.11}$$

where

$$\begin{aligned} E[m_1, pk]^{m_2} &= (g^{m_1} r_1^n)^{m_2} (\text{mod } n^2) \\ &= (g^{m_1 \cdot m_2}(r_1^{m_2})^n) (\text{mod } n^2) \\ &= E[m_1 \cdot m_2, pk]. \end{aligned} \tag{2.12}$$

Security. The Paillier encryption scheme is semantically secure based on hardness computation of the decisional composite residuosity (DCR) assumption. Let N and z be a composite and an integer. It is hard to check z is a N-residuo modulo N^2 as follows,

$$z = y^n \ (\text{mod } n^2), \tag{2.13}$$

where y is kept secret. Thus, DCR problem is intractable. The scheme also achieves security against chosen-plaintext attacks (IND-CPA); encrypted messages are indistinguishable. However, the scheme has malleability which it can not protect against adaptive chosen-ciphertext attacks (IND-CCA2). In certain secure applications such as secure voting and threshold cryptosystems, malleability is a desired property. An improved Paillier cyrptosystem (Paillier and Pointcheval, 1999) has been proposed to address the above limitation.

Threshold Homomorphic Encryption. (Cramer et al., 2001) propose a homomorphic threshold cryptosystem that creates a Boolean circuit to compute secure functions. The properties of the homomorphic threshold cryptosystem are in the following,

- **Addition of plaintexts:** The cryptosystem can easily compute $\bar{a} + \bar{b}$, where $\bar{a}$ and $\bar{a}$ are the encryptions of the plaintexts a and b, respectively.
- **Multiplication by a constant:** The cryptosystem can easily compute $\bar{a} \cdot \alpha$, where $\bar{a}$ is the encryption of the plaintext a and $\alpha \in \mathbb{Z}_n$ is a constant integer.
- **Proving knowledge of plaintext:** Zero-knowledge proof can be used to prove any encryption of the plaintext.
- **Proving knowledge of multiplication:** Zero-knowledge proof can be used to prove the encryption of $\bar{a} \cdot \alpha$, where $\bar{a}$ is the encryption of the plaintext a and $\alpha \in \mathbb{Z}_n$ is a constant integer.
- **Threshold decryption:** Given a plaintext a that is encrypted to $\bar{a}$ by a corresponding public key, each party can use its own private key to decrypt $\bar{a}$. The sharing of partial decryptions allows that each party can securely compute a.

(Damgård and Jurik, 2001) first proposed a generalization of the Paillier encryption scheme to allow the ciphertext in the range of $\mathbb{Z}_{n^{s+1}}$, where $s \geq 1$. They then extend the Paillier cryptosystem based on (Cramer et al., 2001) to a (t, c)-threshold cryptosystem, where c is the total number of parties in the cryptosystem, and t ($1 \leq t \leq c$) is a threshold. Let (pk, sk) be a public and private key pair of the Paillier cryptosystem. (Damgård and Jurik, 2001) applies the efficient secret sharing (Frankel et al., 1998) to decompose sk into c shares, and distributes each share to one and only one party. The secret sharing ensures that only t or more parties working together can recover sk (or decrypt a ciphertext generated by pk). Once the shares are generated, it is safe to delete sk. Some other variants of the Paillier (t, c)-threshold cryptosystem include (Baudron et al., 2001; Fouque et al., 2000; Hirt and Sako, 2000).

2.1.2 Goldwasser-Micali Encryption Scheme

The Goldwasser-Micali (GM) encryption scheme (Goldwasser and Micali, 1982) is a public-key encryption (i.e., a public key and a private key). The scheme can determine whether

a value x is a square mod N using the factorization (p, q) of N as follows,

$$x_p = x(\text{mod } p) \mapsto x_p^{\frac{p-1}{2}} = 1(\text{mod } p) \tag{2.14}$$

$$x_q = x(\text{mod } q) \mapsto x_q^{\frac{q-1}{2}} = 1(\text{mod } q), \tag{2.15}$$

where x that meets above two conditions is a quadratic residue mod N. In the following, we use two parties (i.e., Alice and Bob) to discuss the scheme.

Key Generation. The GM scheme generates keys in a similar way as in the RSA cryptosystem (Rivest et al., 1978). Alice first generates two large prime numbers p and q which both are independent of each other. She then computes $N = p \cdot q$ and finds some non-residue a,

$$a_p^{\frac{p-1}{2}} = -1(mod\ p), a_q^{\frac{q-1}{2}} = -1(mod\ q), \tag{2.16}$$

where the secret key is generated from the factorization (p, q) and the public key is formed of (a, N).

Encryption. Bob has a message m and he sends m to Alice. m can be represented in a string of bits $(m_1, \cdots, m_l)$. Bob first generates a random value b_i for each bit m_i, such that

$$c_i = b_i^2 \cdot a^{m_i}(mod\ N), \tag{2.17}$$

where b_i is generated from modulo N or $gcd(b_i, N) = 1$. Bob sends the encrypted message $(c_1, c_2, \cdots, c_l)$ to Alice.

Decryption. Alice receives the encrypted message $(c_1, c_2, \cdots, c_l)$ from Bob. Alice uses the factorization (p, q) to determine each of $(c_1, c_2, \cdots, c_l)$ that is a quadratic residue;

$$\forall_i^n\ m_i = \begin{cases} 0, & \text{if } c_i \text{ is a quadratic residue,} \\ 1, & \text{otherwise.} \end{cases} \tag{2.18}$$

The outputs of Alice are the plaintext $(m_1, \cdots, m_l)$.

Homomorphic Property. Let c_0 and c_1 be encrypted bits of m_1 and m_2 respectively. We compute c_0 and c_1,

$$c_0 = b_0^2 \cdot a^{m_0} (mod\ N), c_1 = b_1^2 \cdot a^{m_1} (mod\ N). \tag{2.19}$$

Then $c_0 \cdot c_1$ is

$$\begin{aligned} c_0 \cdot c_1 &= (b_0^2 \cdot a^{m_0}) \cdot (b_1^2 \cdot a^{m_1}) (mod\ N) \\ &= (b_0 b_1)^2 \cdot a^{m_0 + m_1} (mod\ N), \end{aligned} \tag{2.20}$$

where $m_0 + m_1 = m_0 \oplus m_1$ (i.e., exclusive-OR $\oplus$ for addition modulo 2). It is easily verified that $c_0 \cdot c_1 (mod\ N)$ is an encryption of 0 if $m_0 = 1$ and $m_1 = 1$ (i.e., $m_0 \oplus m_1$). Thus, $c_0 \cdot c_1$ is the encryption of $m_0 \oplus m_1$.

Security. The GM scheme is semantically secure based on hardness to determine the quadratic residuosity problem modulo $N = p \cdot q$ (composite) where both p and q are large prime numbers. As given the factorization of N, we can solve the quadratic residue problem. The disadvantage of the GM scheme is that it is required to encrypt every single bit of a message which each bit is approximately equivalent to the size of $|N|$. As a result, the size of ciphertext is much larger (e.g., several hundred times) than the initial message. Moreover, in the GM scheme, N needs to be set at least several hundred bits to prevent factorization attacks. The GM scheme is not a practical and efficient cryptosystem.

2.1.3 ElGamal Encryption Scheme

The ElGamal encryption scheme (Gamal, 1985) is a probabilistic public key encryption. Any cylic group G can define an encryption scheme based on the Diffie-Hellman key exhange. Basically the security of the scheme is based on hardness to compute the discrete logarithms of a certain problem in G. In the following, we use two parties (i.e., Alice and Bob) to discuss the scheme.

Key Generation. Alice uses a generator g to generate an information, with order q, of a cyclic group G. She computes

$$y = g^x, \tag{2.21}$$

where $x \in \{1, \cdots, q-1\}$ is a random number. Alice keeps x as the secret key and discloses the public key which contains y, q, g, and the information of G.

Encryption. Bob has a message m with the public key (G, q, g, y) of Alice. He first computes c_1 and s as follows,

$$c_1 = g^r, s = y^r \tag{2.22}$$

where $r \in \{1, \cdots, q-1\}$ is a random number and s is the shared secret. Bob then encrypts the message m by converting m to m' which is an element in G (i.e., $m' \in G$),

$$c_2 = m' \cdot s. \tag{2.23}$$

The ciphertext (c_1, c_2) (i.e., $(g^r, m' \cdot y^r)$) is sent to Alice. In a case, one knows m' that can be used to find y^r. To improve security of the scheme, r should be generated for every new message.

Decryption. Alice receives the ciphertext (c_1, c_2). To recover the message m, Alice first computes the shared secret,

$$\mu = c_1^x. \tag{2.24}$$

and then apply μ to c_2 to recover m,

$$\begin{aligned} c_2 \cdot \mu^{-1} &= (m' \cdot s) \cdot c_1^x \\ &= m' \cdot y^r \cdot g^{-xr} = m' \cdot g^{xr} \cdot g^{-xr} = m' \end{aligned} \tag{2.25}$$

where μ^{-1} is the inverse of μ in the group G. Note that μ^{-1} is the modular multiplicative inverse of μ in the group G/H which H is the multiplicative group of integer modulo n. Alice can convert m' into the message m.

In a general, the ELGamal encryption scheme generates an expansion ratio 2:1 from plaintext message to ciphertext; the scheme can encrypt a single plaintext to many different ciphertexts. The scheme needs two exponentiations in encryption and one exponentiation in decryption.

Homomorphic Property. Let (c_{11}, c_{12}) and (c_{21}, c_{22}) be two encrypted messages of m_1 and m_2 respectively, under the ELGamal encryption scheme, such that

$$(c_{11}, c_{12}) = (g^{r_1}, m_1 \cdot y^{r_1}), (c_{21}, c_{22}) = (g^{r_2}, m_2 \cdot y^{r_2}), \tag{2.26}$$

where both r_1 and r_2 are random numbers (i.e, $r_1, r_2 \in \{1, 2, \cdots, q-1\}$) and $m_1, m_2 \in G$. We can compute

$$\begin{aligned}(c_{11}, c_{12}) \cdot (c_{21}, c_{22}) &= (c_{11} \cdot c_{21}, c_{12} \cdot c_{22}) \\ &= (g^{r_1} \cdot g^{r_2}, (m_1 \cdot y^{r_1}) \cdot (m_2 \cdot y^{r_2})) \\ &= (g^{r_1+r_2}, (m_1 \cdot m_2) \cdot y^{r_1+r_2}).\end{aligned} \tag{2.27}$$

Above Equation 2.27 has showed that $(c_{11}, c_{12}) \cdot (c_{21}, c_{22})$ is the encryption of $m_1 \cdot m_2$.

Security. The decisional Diffe-Hellman (DDH) assumption (Boneh, 1998) is a computational hardness assumption on computing a certain problem that involves discrete logarithms in cyclic group G. Let g be a generator in the multiplicative cyclic group G of order q. In the assumption, as given g^a and g^b $(a, b \in \mathbb{Z}_q)$ that are uniformly and independently selected, it states that g^{ab} is like a random value in G. Intuitively the hardness assumption is based on below two probability distributions that are computationally indistinguishable.

$$\mathbb{P}_1 = (g^a, g^b, g^{ab}), \mathbb{P}_2 = (g^a, g^b, g^c), \tag{2.28}$$

where a, b, and c are randomly and independently selected from $\mathbb{Z}_q$. The ElGamal encryption scheme based on the DDH assumption is semantically secure. However, the encryption scheme is unconditionally malleable that an adversary can possibly transform a ciphertext of the plaintext into another ciphertext of the similar plaintext (chosen-ciphertext attack). Two improved variants of ElGamal encryption scheme, Cramer-Shoup cryptosystem (Cramer and Shoup, 1998) and DHAES (Abdalla et al., 1999), can provide semantic security against the chosen-ciphertext attack.

2.1.4 Boneh-Goh-Nissim Encryption Scheme

Boneh-Goh-Nissim (BNG) encryption scheme (Boneh et al., 2005) allows both addition and multiplication operations with a fixed size of ciphertext. In the scheme, it makes

multiplication possible where pairings can be specified for elliptic curve. A pairing is a map, such that

$$e : G_1 \times G_2 \mapsto G_\tau, \tag{2.29}$$

where both G_1 and G_2 are additive groups, and G_τ is a multiplicative group. For a generator $g \in G$, $P \in G_1$, and $Q \in G_2$, $e(P, Q)$ is equivalent to $e(Q, P)$,

$$e(P, Q) = e(g^p, g^q) = e(g, g)^{pq} = e(g^q, g^p) = e(Q, P), \tag{2.30}$$

where p and q are some integers, $P = g^p$ and $Q = g^q$.

Key Generation. BNG generates a quintuple (q_1, q_2, G, G_1, e) with a security parameter $\lambda \in \mathbb{Z}^+$, where q_1 and q_2 are two different large prime numbers, G is a cyclic group of order $q_1 \cdot q_2$, and e is a paring map from $G \times G$ to G_1 (i.e., $e : G \times G \mapsto G_1$). The scheme then selects two random generators g and u from G and sets $N = q_1 \cdot q_2$ and $h = u^{q_2}$. Note that random generator h is the subgroup of G of order q_1. The public key is $PK = (N, G, G_1, e, g, h)$ and the secret key is $SK = q_1$ of BNG encryption scheme.

Encryption. A message m consists of integers $\{0, 1, \cdots, T\}$, where $T < q_2$. For a simplicity, we set T to 1 here. Using the public key PK, m is encrypted to

$$C = g^m h^r \in G, \tag{2.31}$$

where random number $r \in \{1, 2, \cdots, N\}$ is selected.

Decryption. The plaintext message m is recovered using the private key $SK = q_1$ of the scheme. The ciphertext C is decrypted by

$$C^{q_1} = (g^m h^r)^{q_1} = (g^{q_1})^m. \tag{2.32}$$

Discrete logarithm of C^{q_1} to the base g^{q_1} can be computed. The computation time is $O(\sqrt{T})$ $(0 \leq m \leq T)$ using Pollard's lambda method (Menezes et al., 1996).

Homomorphic Property. Let m_1 and m_2 be two messages and a public key PK be (N, G, G_1, e, g, h). The messages m_1 and m_2 are encrypted to $C_1 = g^{m_1}h^{r_1}$ and $C_2 = g^{m_2}h^{r_2}$ respectively, where $C_1, C_2 \in G$ and $m_1, m_2 \in \{0, 1, \cdots, T\}$.

The product of C_1 and C_2 is the encryption of $m_1 + m_2$,

$$\begin{aligned} C &= C_1 C_2 h^r \\ &= (g^{m_1}h^{r_1})(g^{m_2}h^{r_2})h^r = g^{m_1+m_2}h^{r_1+r_2+r}, \end{aligned} \tag{2.33}$$

where $r_1, r_2, r \in \{1, 2, \cdot, N-1\}$ are random numbers. To get an encryption of $m_1 m_2 (\mathrm{mod} N)$, the scheme first defines two bilinear maps, $g_1 = e(g, g)$ and $h_1 = (g, h)$ where g_1 is of order N and h_1 is of order q_1. Let $\alpha \in \mathbb{Z}$ be some unknown and $r \in \mathbb{Z}_N$ be a random number. Next compute

$$\begin{aligned} C &= e(C_1, C_2)h_1^r \in G_1 \\ &= e(g^{m_1}h^{r_1}, g^{m_2}h^{r_2})h_1^r = e(g^{m_1+\alpha q_2 r_1}, g^{m_2+\alpha q_2 r_2})h_1^r \\ &= e(g, g)^{(m_1+\alpha q_2 r_1)(m_2+\alpha q_2 r_2)}h_1^r = e(g, g)^{m_1 m_2+\alpha q_2(m_1 m_2+m_2 r_1+\alpha q_2 r_1 r_2)}h_1^r \\ &= e(g, g)^{m_1 m_2}h_1^{r+m_1 m_2+m_2 r_1+\alpha q_2 r_1 r_2}, \end{aligned} \tag{2.34}$$

where $r + m_1 m_2 + m_2 r_1 + \alpha q_2 r_1 r_2$ is uniformly distributed in $\mathbb{Z}_N$. The encryption C of $m_1 m_2 (\mathrm{mod}\ N)$ is also uniformly distributed where C is in G_1 (i.e. $C \in G_1$). Thus, the BGN scheme is additively homomoprhic in G_1 for the encryption of the product m_1 and m_2 (i.e., $m_1 m_2 (\mathrm{mod}\ N)$).

Security. The BGN scheme is semantically secure based on hardness computation of the subgroup decision (SD) problem (Boneh et al., 2005). Let p and q be two distinct prime numbers, and $g_p \in G_p$ and $g \in G$ be two generators where the group G is of composite order $n = pq$. The SD problem states that it is hard to distinguish whether a random element x is from either the subgroup G_p or the full group G. An extensive study on the SD problem can be found in (Freeman, 2010; Gjøsteen, 2005)

2.2 Fully-Homomorphic Encryption (FHE)

In fully-homomorphic encryption, it allows to perform more than one operation (addition or multiplication) of semi-homomorphic encryption. Let P and C be the plaintext space and ciphtertext space, K be the key spaces, and E[.] and D[.] be encryption function and decryption function. The encryption scheme is (P, C, K, E, D). Two rings are formed from the plaintexts and the ciphertexts respectively; a plaintext ring $(P, \oplus_p, \otimes_p)$ and a ciphertext ring $(C, \oplus_c, \otimes_c)$. The encryption function $E[.]$ maps the ring P to the ring C (i.e., $E_k : P \mapsto C$ where $k \in K$ is either a public key or a secret key), such that

$$E_k[a] \oplus_c E_k[b] = E_k[a \oplus_p b], \tag{2.35}$$

$$E_k[a] \otimes_c E_k[b] = E_k[a \otimes_p b], \tag{2.36}$$

where a and b are plaintexts (i.e., $a, b \in P$).

The first breakthrough solution (Gentry, 2009) of fully-homomorphic encryption was proposed in 2009. In the solution, noise added to encrypted data grows as the number of addition and multiplication operations increases. As a result, the encrypted data is indecipherable. Gentry resolves the noise issue by modifying the existing scheme to bootstrappable. He converts the bootstrappable somewhat encryption scheme into a fully-homomorphic encryption via a recursive self-embedding. The security of Gentry's scheme is based on hardness of two following problems: certain worse-case problems over ideal lattices and the sparse subset sum problem.

However, in Gentry' scheme, the computation time and ciphertext size increase significantly as the security level increases. Therefore the scheme is impractical to apply in many applications. Since then, many improved Gentry's schemes have been proposed (Brakerski et al., 2012; Coron et al., 2013, 2012; van Dijk et al., 2010; Gentry and Halevi, 2011; Smart and Vercauteren, 2010; Stehlé and Steinfeld, 2010). In this section, we discuss one of the improved schemes, secret key somewhat homomorphic encryption over integers. (van Dijk et al., 2010).

Secret Key Somewhat Homomorphic Encryption over Integers. The scheme consists of three main functions: key generation, encryption and decryption.

Key Generation. Let secret key k_s be an odd integer. k_s is selected from some interval $p \in [2^{\eta-1}, 2^{\eta}]$ where η is a positive integer.

Encryption. Given a bit $m \in \{0, 1\}$, m is encrypted to

$$c = k_s q + 2r + m, \tag{2.37}$$

where the integers q and r are randomly selected from some other defined intervals, and $|2r| < |\frac{k_s}{2}|$. Note that the ciphertext c (residue modulo k_s) and its plaintext have the same parity.

Decryption. The ciphertext c is decrypted to

$$\begin{aligned}(c(\text{mod } k_s))(\text{mod } 2) &= (k_s q + 2r + m(\text{mod } k_s))(\text{mod } 2) \\ &= 2r + m(\text{mod } 2) = m,\end{aligned} \tag{2.38}$$

where the ciphertext $c = k_s q + 2r + m$ and k_s is the secret key.

Homomorphic property. Let $c_1 = k_s q_1 + 2r_1 + m_1$ and $c_2 = k_s q_2 + 2r_2 + m_2$ be two ciphertexts where k_s is the secret key, $q_{\{1,2\}}, r_{\{1,2\}}$ are randomly selected from some other defined intervals, and m_1 and m_2 are plaintexts (i.e., bits). The addition of ciphertexts $c_1 + c_2$ is decrypted to $m_1 + m_2$,

$$c_1 + c_2 = (q_1 + q_2)k_s + 2(r_1 + r_2) + (m_1 + m_2), \tag{2.39}$$

where $r_1 + r_2 < \frac{k_s}{2}$. Thus, $c_1 + c_2$ is equal to

$$((q_1 + q_2)k_s + 2(r_1 + r_2) + (m_1 + m_2)(\text{mod } k_s))(\text{mod } 2) = m_1 + m_2. \tag{2.40}$$

The product of ciphertexts $c_1 \cdot c_2$ is decrypted to $m_1 m_2$,

$$\begin{aligned}c_1 \cdot c_2 =&(k_s q_1 q_2 + 2q_1 r_2 + 2q_2 r_1 + m_1 q_2 + m_2 q_1)k_s \\ &+ 2(2r_1 r_2 + m_1 r_2 + m_2 r_1) + (m_1 m_2),\end{aligned} \tag{2.41}$$

where $2r_1r_2 + m_1r_2 + m_2r_1 < \frac{k_s}{2}$. Thus, $c_1 \cdot c_2$ is equal to

$$\begin{aligned}&(((k_sq_1q_2 + 2q_1r_2 + 2q_2r_1 + m_1q_2 + m_2q_1)k_s \\ &+ 2(2r_1r_2 + m_1r_2 + m_2r_1) + (m_1m_2))(\text{mod } k_s))(\text{mod } 2) = m_1m_2. \end{aligned} \tag{2.42}$$

As fully homomorphic properties described above, we can apply them to evaluate a Boolean function $f(x_1, x_2, \cdots, x_n)$ where $\forall_{i=1}^{n} x_i \in \{0, 1\}$ is a bit that is encrypted to c_i. Let the addition operator be applied to the Boolean function. As the number of additions increases, noise error is increasing that can lead to $r_1 + r_2 > \frac{k_s}{2}$, likewise for the multiplication operator. As the number of multiplications increases, noise error is also increasing that can lead to $2r_1r_2 + m_1r_2 + m_2r_1 > \frac{k_s}{2}$. In both cases above, when the noise error that is greater than $\frac{k_s}{2}$, their ciphertext is indecipherable (i.e., the decryption of $f(c_1, c_2, \cdots, c_n)$ is highly probable different from $f(x_1, x_2, \cdots, x_n)$). Therefore, the scheme can support only low-degree Boolean functions on encrypted data. This limitation is a reason that the scheme is called, somewhat homomoprhic encryption scheme.

Security. The security of somewhat homomoprhic encryption scheme over integers can be reduced to the hardness of the great common divisor (GCD) problem in approximating integer. (Howgrave-Graham, 2001). Let a and b be two input integers. Computation of $gcd(a, b) = d$, is in polynomial time. If d is sufficiently large then it may possibly give some additive errors on either of a and b, or both. CGD can still recover the result regardless of the error. This example is known as the approximate integer common divisor problem. Let $\forall_{i=1}^{n} u_i$ be near multiples where $u_i = pq_i + r_i$. To find p, or at least $p \cdot gcd(q_1, q_2, \cdots, q_n)$ in the near multiples $u_1, u_2, \cdots, u_n$, it is computational hardness in the encryption scheme when p is relatively small as compared to the size of the near multiples.

In conclusion, many fully-homomorphic encryption schemes (Yi et al., 2014) are still underperformed than the semi-homomorphic encryption schemes. Therefore, we mainly focus to investigate semi-homomorphic encryption in our thesis.

2.3 Oblivious Transfer

Oblivous Transfer (OT) is a basic primitive in secure computation. The basic idea of OT is that a sender uses OT protocol to send information to the receiver, but remains oblivious to

Protocol 1: One-Out-Of-Two Oblivious Transfer Protocol

Input: Alice has an input bit σ and Bob has two inputs, B_0 and B_1 .
Output: Alice learns B_σ

1 Bob generates a random number $C \in 1, \cdots, p-1$ which is sent to Alice along with p and a generator g.
2 Alice selects a random number $k \in 1, \cdots, p-1$, and then sets $P_\sigma = g^k$ and $P_{1-\sigma} = \frac{C}{P_\sigma}$. Alice sends P_0 to Bob.
3 Bob evaluates $\frac{C}{P_0}$ to get P_1 and then creates $E_0 = (g^{r_0}, H((P_0)^{r_0})) \oplus B_0$ and $E_1 = (g^{r_1}, H((P_1)^{r_1})) \oplus B_1$, where r_0, r_1 are randomly selected from $1, \cdots, p-1$. He sends E_0, E_1 to Alice.
4 Alice computes $H((P)^{r_\sigma}) = H((g^{r_\sigma})^k)$ to get B_σ.

what information is received. For simplicity, we use the one-out-of-two oblivious protocol (O_2^1) to describe the details of OT. The one-out-of-two oblivious protocol (O_2^1) is the two-party protocol (e.g., Alice and Bob). Alice has an input bit σ and Bob has inputs B_0 and B_1. The protocol allows Alice learns only B_σ and Bob learns nothing about σ. (Even et al., 1985) propose the one-out-of-two oblivious protocol (O_2^1) by generalizing Rabins "oblivious transfer" (Rabin, 1981).

(Naor and Pinkas, 1999) propose the one-out-of-two oblivious protocol (O_2^1) that is described in the following. They first use a hash function H to disclose nothing even one of the parties tries to deviate the protocol. All operations except the function H and $\oplus$ (exclusive or) are computed in modulo p.

In the Protocol 1, the choice of Alice (σ) is not revealed to Bob. Bob receives a value in either g^k or C/g^k, where k is selected randomly from the uniform distribution. As both g^k and C/g^k are uniformly distributed from $0, \cdots, p-1$, the received value is a random number to Bob. Thus, Bob learns nothing other than a random number. Alice learns nothing from a random number C, or from inferring encrypted data E_0 or E_1. However, Alice can decrypt E_σ to get the result where she knows the decryption key. Based on the hardness of the Diffie-Hellman assumption (Boneh, 1998), Alice can not decrypt the cipher text without the decryption key. At the end of the oblivious transfer computation, Alice learns B_σ from the inputs of Bob, and Bob learns nothing about the choice of Alice. The oblivious transfer has been extended to support one-out-of-N (O_N^1) and k-out-of-N (O_N^k) cases (Ishai and Kushilevitz, 1997; Brassard et al., 1986).

2.4 Secure Multi-party Computation (SMC)

Secure multi-party computation (SMC) (Goldreich, 2004) is a fundamental field in cryptography. It allows multi-parties to jointly compute a function over their respective inputs, while keeping every input confidential. Let $x_1, x_2, \cdots, x_m$ be the inputs of m parties respectively, and $f : (\{0,1\}^*)^m \rightarrow (\{0,1\}^*)^m$ be an m-ary function. SMC hides all x_i's, but computes $f(x_1, x_2, \cdots, x_m) = \{f_i(x_1, x_2, \cdots, x_m)\}_{i \in \{1,2,\cdots,m\}}$, where $f_i(x_1, x_2, \cdots, x_m)$ is the output to the i-th party.

Therefore, SMC is very effective in protecting personal privacy. It has been applied in privacy-preserving computation in different areas such as in auction, benchmarking and data mining. To make the auction in a fair way, the bid of every bidder should be kept confidential. If the auctioneer can learn the bid of a bidder b_1, he/she can always work with other bidder b_o to force the price that is always below maximum of the bidder b_1. To win the bid, the bidder b_1 is forced to pay more. Many privacy-preserving auction protocols of SMC (Brandt and Sandholm, 2008; McSherry and Talwar, 2007; Malkhi et al., 2004; Pinkas, 2002; Naor et al., 1999) have been proposed to address the issue. In the benchmarking, two companies in the same business line are interested to learn how well each other is doing. Both companies have some parameters to use for benchmarking, such as their employee salaries, the productivity, the profit relative to size and many more. However, many of the such sensitive data are private and confidential which data can not be leaked especially to their competitors. (Kerschbaum, 2008; Kiltz et al., 2005; Atallah et al., 2004; Du and Atallah, 2001b) have proposed privacy-preserving benchmarking protocols to securely compute the statistics of the performance measures of the companies without revealing their private data.

SMC have been extensively applied in privacy-preserving data mining (PPDM) (Lindell and Pinkas, 2008; Verykios, Bertino, Fovino, Provenza, Saygin and Theodoridis, 2004) in the context of multiple parties. These parties are not willing to share their data directly, but would like to work together to learn the output of any agreed mining task. Various SMC solutions have been proposed to serve different mining tasks, such as decision tree (Vaidya, Clifton, Kantarcioglu and Patterson, 2008), Naïve Bayes (Vaidya, Kantarcioglu and Clifton, 2008), support vector machine (Teo et al., 2013; Yu, Jiang and Vaidya, 2006) and singular value decomposition (Han et al., 2009).

2.5 Secure Multi-party Computation Models

In this section, we discuss three models of secure multi-party computation (SMC). The first is Yao's circuit model (Section 2.5.1) and the second is arithmetic circuit model (Section 2.5.2) . These two models use oblivious transfer (Section 2.3) as a basic primitive of their secure computations. The last model of SMC we discuss is the homomorphic cryptography model (Section 2.5.3).

2.5.1 Yao's Circuit Model

(Yao, 1986) proposes a circuit evaluation protocol to be secure against semi-honest adversaries. This protocol is rigorously proven by (Lindell and Pinkas, 2009). Let CRT be a Boolean circuit that accepts two inputs $a, b \in \{0,1\}^*$ and outputs $O_{CRT}(a,b) \in \{0,1\}^*$. In the evaluation, O_{CRT} can always be found such that $O_{CRT}(a,b) = f(a,b)$ for any deterministic functionality $f : \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1\}^*$.

We first show to use a single gate that has 2-input wires (i.e., input a of Alice and b of Bob) on Yao's protocol. As the Boolean circuit, the gate can represent a function $g : \{0,1\} \times \{0,1\} \mapsto \{0,1\}$ with two input wires ω_1 and ω_2, and one output wire ω_3. One party uses a key generation algorithm $K(1^N)$ to generate six random and independent keys $k_1^0, k_1^1, k_2^0, k_2^1, k_3^0, k_3^1$ of length N, where k_x^y is a garble value of ω_x's value y. The party then can generate four values as follows,

$$
\begin{aligned}
crt_{0,0} &= E_{k_1^0}(E_{k_2^0}(k_1^{g(0,0)})) \\
crt_{0,1} &= E_{k_1^0}(E_{k_2^1}(k_1^{g(0,1)})) \\
crt_{1,0} &= E_{k_1^1}(E_{k_2^0}(k_1^{g(1,0)})) \\
crt_{1,1} &= E_{k_1^1}(E_{k_2^1}(k_1^{g(1,1)})),
\end{aligned}
$$

where E stands for private key encryption. The encryption scheme ensures that all ciphertexts are semantic security (i.e., computational indistinguishability). Another property of the encryption scheme is allowed to verify that a ciphertext is in the range of that a given key. A random permutation of the above four values is formed a garbled table of the gate g. Alice first sends k_1^α and the garbled table to Bob where α is a private input of Alice. Then Alice securely transfers the key k_2^β to Bob by 1 out of 2 oblivious transfer

OT_1^2 protocol (Even et al., 1985). In the end of protocol execution, Alice learns nothing from which key k_2^y is selected and Bob learns only key k_2^{1-y}. Bob can decrypt the garbled table values with the keys k_1^α and k_2^β. The private encryption scheme can decrypt one of four values in the range with a negligible probability. Thus Bob learns the output key $k_3^{\alpha,\beta}$.

In a case such that no input wire for the gate g, Bob can still decrypt the four values by two input keys to get the output key of g after receiving the garbled table from Alice. To get a result of $CRT(a, b)$ from the gates, by assuming that output wire u carries the value of $CRT(a, b)$, Alice first sends the decryption table that contains $(0, k_u^0)$ and $(0, k_u^1)$ to Bob. Bob knows the output key that can be the result 0 or 1 from the output wire u. Therefore, Bob can securely compute $CRT(a, b)$.

In Yao's protocol, the construction of the garbled table (Lindell and Pinkas, 2000) involves one 1-out-of-2 oblivious transfer OT_1^2 [1] (Naor and Pinkas, 2001) per each input bit. Each gate needs four times of pseudo-random function evaluations. Given the functionality f on L gates with N-bits length of inputs, the most communication cost is the pseudo-random function that generates $4L$ times of length of the random keys from one party to the other. In term of computation complexity, it needs N times to invoke OT_1^2 protocol between two parties and $4L$ times to run the pseudo-random function. This makes Yao's protocol infeasible for some complex functions that require many either gates or data inputs.

Recently, (Lindell and Pinkas, 2007) combine the cut-choose approach and some other new techniques to efficiently prove consistency of inputs on Yao's circuit evaluation protocol targeted on two malicious parties. FairPlay (Ben-David et al., 2008) uses Yao's circuit evaluation protocol (Yao, 1986) to compile a generic function into a Boolean circuit.

2.5.2 Arithmetic Circuit Model

Arithmetic circuit (Goldreich, 2004) that contains addition and multiplication gates over Galois Field of size two (i.e. $GF(2)$) can securely compute any deterministic function. The basic idea of the arithmetic circuit is to propagate and split shares of the input wires of the circuit into shares of all wires of the circuit via secure computation. In the end of

[1] OT_1^2 is considered a special case of 1-out-of-n oblivious transfer OT_1^n protocol (Naor and Pinkas, 1999, 2001) by which a sender has n messages and receiver selects a message out of n that is unknown to the sender.

execution, the output wires of the circuit are split into shares held by each participating party.

In the arithmetic circuit, multiplication gates incur the expensive cost in which each multiplication gate needs to invoke OT_1^4 protocol. The overall performance of the multiplication gate is improved (Naor and Pinkas, 1999) by running OT_1^n in $\log N$ parallel invocations of OT_1^2 protocol. However, the computation overhead in oblivious transfer is obviously more costly than the communication overhead. To address this issue, (Naor and Pinkas, 2001) propose an oblivious protocol based on the decisional Diffie-Hellman that assumes a receiver only performs two exponentiations and a sender only performs $O(n)$ exponentiations. The communication overhead of the (Naor and Pinkas, 2001) protocol is $O(n)$. (Ben-Or et al., 1988) apply some non-cryptographic techniques to represent f as an arithmetic circuit in the party setting of an honest majority. In the non-cryptographic approach, the arithmetic model can be evaluated using the weaker computational models of formulas (Bar-Ilan and Beaver, 1989) over large fields.

All data or variables are Boolean values in the arithmetic circuit. Let a_1 and a_2 be inputs of Alice, and b_1 and b_2 be inputs of Bob. A circuit gate has two inputs of a wire such that the first input is a_1 and b_1 and the second is a_2 and b_2. In the addition computation, Alice sets the output wire of the gate to $a_1 + a_2$ and Bobs sets the output wire to $b_1 + b_2$. In the multiplication computation, Alice and Bob invoke the oracle using Equation 2.43. In the end of execution, Alice and Bob get a_3 and b_3 respectively where $a_3 + b_3$ is the output of the multiplication gate.

$$(a_1 + b_1) \times (a_2 + b_2) = a_3 + b_3. \tag{2.43}$$

Secure Computation: $a_3 + b_3 = (a_1 + b_1) \times (a_2 + b_2)$ can be transformed to $b_3 = (a_1 + b_1) \times (a_2 + b_2) + a_3$ for Boolean values. Alice first selects $a_3 \in \{0, 1\}$ to compute the following four terms from $(a_1 + b_1) \times (a_2 + b_2) + a_3$ which corresponds to $(b_1, b_2) \in ((0,0),(0,1),(1,0),(1,1))$, such that (1) $a_1 \times a_2 + a_3$, (2) $a_1 \times (a_2 + 1) + a_3$, (3) $(a_1 + 1) \times a_2 + a_3$, and (4) $(a_1 + 1) \times (a_2 + 1) + a_3$. As Bob knows b_1 and b_2, he can use 1-out-of-4 oblivious transfer OT_1^4 protocol to get $2b_1 + b_2 + 1 \in \{1, 2, 3, 4\}$-th element of previously generated terms of Alice and learns nothing else. Bob can get b_3 (i.e., $(a_1 + b_1) \times (a_2 + b_2) + a_3$). Alice and Bob each hold their private outputs a_3 and b_3 respectively.

2.5.3 Homomorphic Cryptography Model

Homomorphic cryptography model uses the homomorphic encryption protocol that is very effective in protecting privacy in secure multi-party computation. Homomorphic encryption allows specific types of operations to perform on ciphertexts (i.e., encrypted messages). The encrypted result of the operations, when decrypted, is equivalent to the result of operations performed on the messages (i.e., plaintexts). For example, Paillier Cryptosystem, (Section 2.1) is a public key cryptosystem with the additive homomorphic property. Let (pk, sk) be a public/private key pair generated by two prime numbers p and q, and m_1 and m_2 be plaintexts. Let $E[.]$ and $D[.]$ be the encryption and decryption functions respectively, and $n = pq$. Then, the product of ciphertexts of m_1 and m_2 decrypts to m_1+m_2, i.e. $D[sk, (E[pk, m_1] \cdot E[pk, m_2] \bmod n^2)] = (m_1 + m_2) \bmod n$. The ciphertext of m_1 raised to the power of m_2 decrypts to $m_1 \cdot m_2$, i.e., $D[sk, (E[pk, m_1]^{m_2} \bmod n^2)] = (m_1 \cdot m_2) \bmod n$. Such the above property allows the homomorphic encryption protocol that can compute more complex functions than in Yao's circuit model (Section 2.5.1) and in the arithmetic circuit model (Section 2.5.2). The details of homomorphic encryption can be found in Sections 2.1 and 2.2.

2.6 Two Adversary Models of SMC

An SMC protocol that can withstand attacks from adversaries is secure. We use the definitions of two adversarial behavior models of secure multi-party computation (SMC): semi-honest adversaries (Lindell and Pinkas, 2008) and malicious adversaries (Lindell and Pinkas, 2008). Thus, the SMC protocol can be proven secure (i.e. privacy protection) under the adversarial model (by knowing adversary power).

- **Semi-honest adversaries:** In the case of the two-party model, Alice and Bob strictly follow the protocol, but attempt to infer the information received by them during the protocol execution. This model may be considered a realistic model (Aggarwal and Yu, 2008) of adversarial behavior in many real world applications.

- **Malicious adversaries:** In the case of the two-party model, Alice and Bob can deviate arbitrarily from the protocol such as aborting the protocol prematurely and

learning information received from each other by substituting their inputs. Zero-knowledge proof (Goldwasser et al., 1989) that can prove assertions made by one party without revealing additional information is a technique to keep the malicious party from the protocol deviation.

In the security proof of an SMC protocol, indistinguishability assumes that an adversary can not distinguish ciphtexts based on the messages he/she encrypts or the distributions of two random variables. We discuss two kinds of the indistinguishability: computational indistinguishability and statistical indistinguishability.

Computational Indistinguishability. The hardness of solving some problems (e.g., NP-hard problems) can also be used to determine if the distributions of two random variables are indistinguishable.

Definition 2.1 *(Agrawal et al., 2003) Let X and Y be two random variables over a finite set $\Omega_z = \{0,1\}^z$. Let $\mathcal{A}_z$ be a probabilistic algorithm, which outputs either 0 or 1 given a value in Ω_z. We say that the distribution of X is computationally indistinguishable from that of Y, if for any probabilistic polynomial-time algorithm $\mathcal{A}_z$, any positive polynomial $p(z)$, and all sufficiently large z's*

$$|\Pr[\mathcal{A}_z(X) = 1] - \Pr[\mathcal{A}_z(Y) = 1]| < \frac{1}{p(z)}.$$

Statistical Indistinguishability. The statistical distance can be used to determine if the distributions of two random variables are indistinguishable.

Definition 2.2 *(Goldreich, 2001) Let X and Y be two random variables over a finite set $\Omega_z = \{0,1\}^z$. Then X and Y is indistinguishable if their statistical distance is negligible if for every positive polynomial $p(z)$ and all sufficiently large z's, their statistical distance is negligible. That is,*

$$\frac{1}{2}\sum_{a\in\Omega_z} |Pr[X = a] - Pr[Y = a]| < \frac{1}{p(z)}. \tag{2.44}$$

By the Definition 2.2 above, X and Y are then computational indistinguishable if their distributions are statistically indistinguishable.

In the following, we will discuss each adversarial model in detail: privacy w.r.t semi-honest behavior in Section 2.6.1 and privacy w.r.t malicious behavior in Section 2.6.2.

2.6.1 Privacy w.r.t Semi-Honest Behavior

(Goldreich, 2004) gives the definition of the semi-honest behavior below.

Definition 2.3 *(privacy w.r.t semi-honest behavior): Let* $f : \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1\}^* \times \{0,1\}^*$ *be a functionality, where* $f_1(a,b)$ *(resp.,* $f_2(a,b)$*) denotes the first (resp., second) element of* $f(a,b)$*. Let* Π *be a two-party protocol for computing* f*. The* ***view*** *of the first (resp., second) party during an execution of* Π *on* (a,b)*, denoted* $\mathbf{VIEW}_1^{\Pi}(a,b)$ *(resp.,* $\mathbf{VIEW}_2^{\Pi}(a,b)$*), is* $(a, r, m_1, \ldots, m_t)$ *(resp.,* $(b, r, m_1, \ldots, m_t)$*), where* r *represents the outcome of the first (resp., second) party's internal coin tosses, and* m_i *represents the* i^{th} *message it has received. The* ***output*** *of the first (resp., second) party after the execution of* Π *on* (a,b)*, denoted* $\mathbf{OUTPUT}_1^{\Pi}(a,b)$ *(resp.,* $\mathbf{OUTPUT}_2^{\Pi}(a,b)$*), is implicit in the party's own view of the execution, and* $\mathbf{OUTPUT}^{\Pi}(a,b) = (\mathbf{OUTPUT}_1^{\Pi}(a,b), \mathbf{OUTPUT}_2^{\Pi}(a,b))$.

- ***Deterministic case:*** *For a deterministic functionality* f*,* Π *can privately compute* f *if there exists probabilistic polynomial-time algorithms of* S_1 *and* S_2*, such that*

$$\{S_1(a, f_1(a,b))\}_{a,b\in\{0,1\}^*} \stackrel{c}{\equiv} \{\mathbf{VIEW}_1^{\Pi}(a,b)\}_{a,b\in\{0,1\}^*}$$
$$\{S_2(b, f_2(a,b))\}_{a,b\in\{0,1\}^*} \stackrel{c}{\equiv} \{\mathbf{VIEW}_2^{\Pi}(a,b)\}_{a,b\in\{0,1\}^*},$$

  where $|a| = |b|$ *and* $\stackrel{c}{\equiv}$ *indicates computational indistinguishability by (non-uniform) families of polynomial-size circuits.*

- ***General case:*** Π *can privately compute* f *if there exists probabilistic polynomial-time algorithms of* S_1 *and* S_2*, such that*

$$\{((S_1(a, f_1(a,b)), f(a,b))\}_{a,b} \stackrel{c}{\equiv} \{(\mathbf{VIEW}_1^{\Pi}(a,b), \mathbf{OUTPUT}^{\Pi}(a,b))\}_{a,b}$$
$$\{((S_2(b, f_2(a,b)), f(a,b))\}_{a,b} \stackrel{c}{\equiv} \{(\mathbf{VIEW}_2^{\Pi}(a,b), \mathbf{OUTPUT}^{\Pi}(a,b))\}_{a,b}.$$

The above definition states that a computation is secure if the views of parties are simulatable on their respective inputs and the protocol outputs to the parties. This model

helps to prevent parties who strictly follow the protocol to learn additional information except their inputs and outputs.

2.6.2 Privacy w.r.t Malicious Behavior

(Kantarcioglu and Kardes, 2006) gives the definition of malicious behavior model based on the existing work of (Cramer et al., 2001). The suggested model is based on the two-party model targeting on the malicious adversary. The complexity of the malicious model is usually greater than the semi-honest model which the malicious model needs to verify each step of execution with the previous computations. To prove security in the malicious model, we can simulate an adversary in the ideal model (i.e., exists a trusted party) that it generates outputs are computational indistinguishable from any adversary in the real-life model.

In this model, private information of an honest party after executing secure building blocks (e.g., secure comparison, secure intersection, and secure scalar product protocol) is guaranteed that it is not learned by the malicious party. The following definitions are from (Kantarcioglu and Kardes, 2006). In the definitions, Π is a two-party protocol. We first discuss the executions of Π in the real life model and the ideal model. To prove the security of Π protocol in the real-life model, we simulate Π in the ideal model with the existing of one trusted party.

Definition 2.4 ***(malicious adversaries in the real-life model)****: Let each party T_i has a secret input u_i^s and a public input u_i^p where $i \in 1, \cdots, n$ and n is the number of participating parties. After the execution of the protocol Π, each party T_i gets a public output v_i^p and a secret output v_i^s. An adversary A collects the public inputs and public outputs from other participating parties. After the execution of the protocol Π, the outputs of the adversary and the party T_i are $\mathbf{ADVR}_{\Pi,A}(k, \vec{u}, C, z, \vec{r})$ and $\mathbf{EXEC}_{\Pi,A}(k, \vec{u}, C, z, \vec{r})$ respectively, where*

- *k is the security parameter,*
- *$\vec{u} = (u_1^p, u_1^s, u_2^p, u_2^s)$ is the inputs of the participating parties,*
- *$C \in \{T_1, T_2\}$ is the corrupted Party,*

- $z \in \{0,1\}^*$ *is the auxiliary input*[2]*, and*
- $\overrightarrow{r} = (r_1, r_2, r_A)$ *is the random inputs of the participating parties and the adversary.*

Let

$$\begin{aligned}\mathbf{EXEC}_{\Pi,A}(k, \overrightarrow{u}, C, z, \overrightarrow{r}) =&(\mathbf{ADVR}_{\Pi,A}(k, \overrightarrow{u}, C, z, \overrightarrow{r})),\\ &\mathbf{EXEC}_{\Pi,A}(k, \overrightarrow{u}, C, z, \overrightarrow{r})_1, \mathbf{EXEC}_{\Pi,A}(k, \overrightarrow{u}, C, z, \overrightarrow{r})_2),\end{aligned}$$

where the random variable is $\mathbf{EXEC}_{\Pi,A}(k, \overrightarrow{u}, C, z)$ *to describe* $\mathbf{EXEC}_{\Pi,A}(k, \overrightarrow{u}, C, z, \overrightarrow{r})$ *(i.e.,* $\overrightarrow{r}$ *is uniformly chosen). Finally, a distribution ensemble* $\mathbf{EXEC}_{\Pi,A}$ *with the security parameter* k *and the index* $(\overrightarrow{u}, C, z)$ *is*

$$\mathbf{EXEC}_{\Pi,A} = \{\mathbf{EXEC}_{\Pi,A}(k, \overrightarrow{u}, C, z)\}_{k\in\mathbb{N}, \overrightarrow{u}\in(\{0,1\}^*)^4, C\in(T_1,T_2), z\in\{0,1\}^*}.$$

Definition 2.5 ***(malicious adversaries in the ideal model)****: Let* $f : \mathbb{N} \times (\{0,1\}^*)^4 \times \{0,1\}^* \mapsto (\{0,1\}^*)^4$ *be a probabilistic two-party function that is computable in probabilistic polynomial time. The output of* f *is given as* $f(k, u_1^p, u_1^s, u_2^p, u_2^s, r) = (v_1^p, v_1^s, v_2^p, v_2^s)$ *where* k *and* r *are the security parameter and the random input respectively. In the ideal model, a trusted party first gathers all the inputs of the parties, computes* f*, and then returns the party* T_i *with the output values* (v_i^p, v_i^s)*. An adversary* S *may replace its private input and public input with other different values. The ideal model is, likewise the real-life model,*

$$\begin{aligned}\mathbf{IDEAL}_{f,S}(k, \overrightarrow{u}, C, z, \overrightarrow{r}) =&(\mathbf{ADVR}_{f,S}(k, \overrightarrow{u}, C, z, \overrightarrow{r})),\\ &\mathbf{IDEAL}_{f,S}(k, \overrightarrow{u}, C, z, \overrightarrow{r})_1, \mathbf{IDEAL}_{f,S}(k, \overrightarrow{u}, C, z, \overrightarrow{r})_2),\end{aligned}$$

where $\mathbf{IDEAL}_{f,S}(k, \overrightarrow{u}, C, z, \overrightarrow{r})$ *is the collection of the outputs. The distribution of* $\mathbf{IDEAL}_{f,S}(k, \overrightarrow{u}, C, z, \overrightarrow{r})$ *is* $\mathbf{IDEAL}_{f,S}(k, \overrightarrow{u}, C, z)$ *where* $\overrightarrow{r}$ *is uniformly distributed. Then, we define a distribution ensemble* $\mathbf{IDEAL}_{f,S}$ *as follows,*

$$\mathbf{IDEAL}_{f,S} = \{\mathbf{IDEAL}_{f,S}(k, \overrightarrow{u}, C, z)\}_{k\in\mathbb{N}, \overrightarrow{u}\in(\{0,1\}^*)^4, C\in(T_1,T_2)z\in\{0,1\}^*}.$$

[2] The auxiliary input is a standard method (tool) to prove the composition theorem. Intuitively, an adversary gathers information from the auxiliary input of other interactions occurring before the current interaction.

In a simulation of the ideal model, the adversary S first sees the u_1^p and the u_2^p and the secret value u_c^s of the corrupted party. Subsequently, the adversary S replaces u_c^p and u_c^s with $\bar{u}_c^p$ and $\bar{u}_c^s$ respectively of its choices. The trusted party uses the modified inputs to evaluate f. In the end of evaluation, the output values (v_i^p, v_i^s) are sent to the party P_i. Again, the adversary can disclose the values v_1^p, v_2^s to the corrupted party.

Definition 2.6 ***(security in the static malicious adversary setting)**: Let f be a two-party function. In the static setting of the two-party protocol Π can securely evaluate f if for any probabilistic polynomial time adversary A, there exists an ideal-model adversary S to run in a polynomial time of A, and such that*

$$\mathbf{IDEAL}_{f,S} \stackrel{c}{\equiv} \mathbf{EXEC}_{\Pi,A},$$

where $\stackrel{c}{\equiv}$ indicates that two ensembles are computational indistinguisable.

In the malicious model, security is proven via a simulation which an adversary in the ideal model that gives the outputs are computational indistinguishable by allowing any adversary in the real-life model. In other words, an ideal model adversary S allows to run on any giving real-life adversary A as a subroutine. This subroutine works in a black-box fashion to show the views (i.e., the views of the ideal model and that of the real-life model) that are computationally indistinguishable.

2.7 Discussion

In secure multi-party computation, a semi-honest adversary strictly follows the protocol and will not collude with others but is interested in inferring additional knowledge using polynomial-time computations. In contrast, a malicious adversary can arbitrarily deviate from the protocol such as altering its input. SMC protocols based on the malicious model require to use expensive methods such as zero-knowledge proof (Goldwasser et al., 1989). The zero-knowledge proof is a method that a prover can prove the giving statement to a verifier.

Even the semi-honest model is much weaker than the malicious model. (Aggarwal and Yu, 2008; Lindell and Pinkas, 2002) suggest that the semi-honest model is more realistic to many real world applications. One of the possible reasons is that deviating from the protocol is a non-trivial task in today complex applications. Another reason is that the

malicious model incurs expensive computation cost as compared to the semi-honest model. Therefore, many SMC protocols based on the semi-honest model have been proposed to allow privacy protection, especially applying in many data mining tasks (Teo et al., 2013; Han et al., 2009; Vaidya, Kantarcioglu and Clifton, 2008; Vaidya, Clifton, Kantarcioglu and Patterson, 2008; Yu, Jiang and Vaidya, 2006). (Han and Ng, 2008) state that it is not possible to have an SMC protocol that can withstand all forms of attacks from the malicious adversary which can arbitrarily deviate the protocol. However, they propose a few methods (Han and Ng, 2008) to withstand some specific attacks from the malicious adversary. In this thesis, we mainly focus on SMC protocols based on the semi-honest model.

Chapter 3

A Survey of Privacy-Preserving Data Mining

In this chapter, we survey the overview of the distributed privacy-preserving data mining. In this thesis, the distributed privacy-preserving data mining is also known as privacy-preserving data mining (PPDM) unless stated otherwise. In recent years, rapid advances in automated data collection tools and data storage technology have led the widespread proliferation of huge amount of distributed data owned by different parties. The distributed data may contain some sensitive information. This raises concerns about privacy protection on the underlying data in data mining. Generally speaking that data mining can be viewed as a threat to privacy violation. Many techniques have been proposed to allow privacy computation in data mining. Thus, distributed privacy preservation is one of active research areas in data mining.

A survey of state-of-the-art privacy-preserving data mining algorithms can be found in (Verykios, Bertino, Fovino, Provenza, Saygin and Theodoridis, 2004). The objective of many privacy-preserving techniques is to design such methods that can allow continuation to be effective in data mining tasks without violating privacy. Many privacy-preserving techniques use some form of transformation on the data to protect data privacy. These techniques minimize privacy leak by reducing the granularity of data representation. However less granular data can incur in some loss of effectiveness (i.e., reduce data utility) in many data mining algorithms. Therefore, it is always a trade-off consideration between information loss and privacy in the design of privacy-preserving techniques.

In many cases, any individual party may wish to collect aggregate results from distributed data which is partitioned across different parties. Data normally can be partitioned into either a horizontal partition (i.e., objects with the same set of attributes are distributed across different parties) or a vertical partition (i.e., different attributes of the same set of objects are distributed across different parties), and both of them. Figure 3.1 shows different types of data partitions for the two-party model (e.g., Alice and Bob)

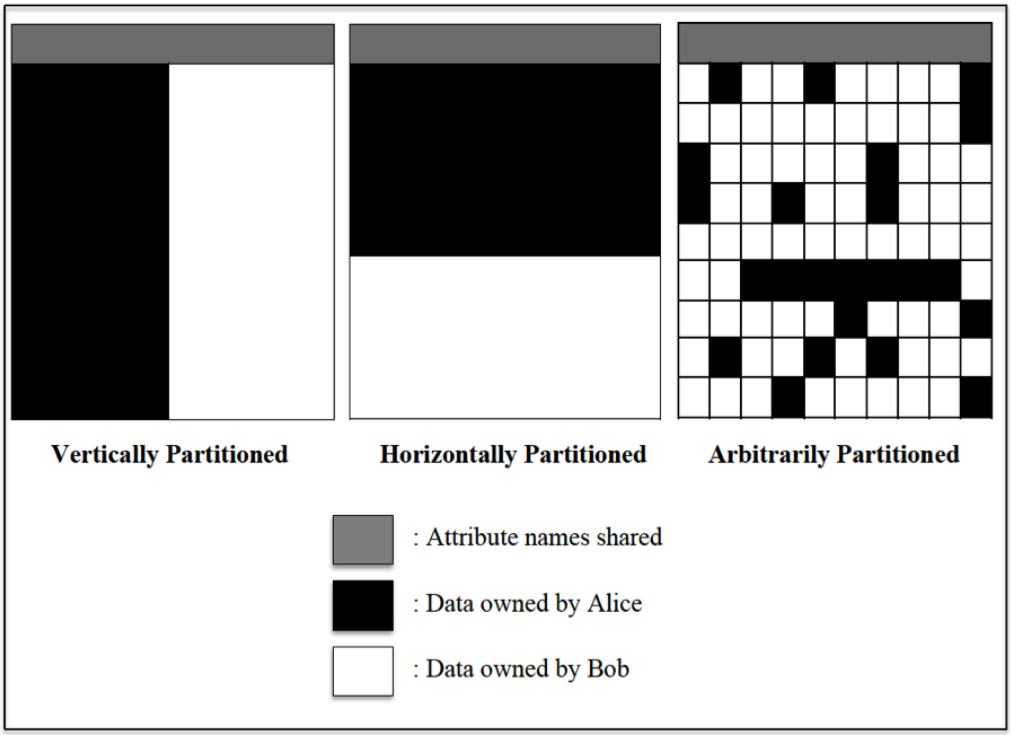


Figure 3.1: Different data partitions on privacy-preserving data mining

in PPDM. In the horizontally partitioned data, Alice and Bob can hold different rows of the same set of attributes in the table. Alice and Bob each hold different columns (i.e., attributes) of the same set of objects in the table for the vertically partitioned data. In the arbitrarily partitioned data, Alice and Bob each can hold any data value in the table.

When participating parties may not want to share their entire data, they may agree to restrict their data sharing with the use of various proposed protocols. In PPDM, many proposed methods maintain data privacy of each participating party, while collecting aggregate results over the distributed data. We next discuss two privacy preservation techniques in PPDM, randomization (Agrawal and Srikant, 2000; Agrawal and Aggarwal, 2001) and secure multi-party computation (SMC) (Lindell and Pinkas, 2000).

Randomization in PPDM. The randomization method is a technique to mask the attribute values of records by adding noise to the data (Agrawal and Srikant, 2000; Agrawal and Aggarwal, 2001). Any individual record normally can not be recovered if noise added to data is sufficiently large. Therefore, many proposed privacy-preserving data mining based on the randomization method are to derive aggregation distributions from the distorted (perturbed) records. Let $X = x_1, x_2, \cdots, x_n$ be a set of data records. In each record $x_i \in X$, a noise component getting from the probability distribution $f_Y(y)$ is added to the

record. Let $y_1, y_2, \cdots, y_n$ be noise components which are randomly and independently from the distribution $f_Y(y)$. Thus, a new set of perturbed records is $x_1+y_1, x_2+y_2, \cdots, x_n+y_n$. Since the added noise to the original records is sufficiently large, the original records cannot be easily inferred from the perturbed records (i.e., the original records cannot be recovered). However, the distribution of the original records can be correctly recovered.

The constructions of decision trees (Agrawal and Aggarwal, 2001; Du and Zhan, 2003), association rule (Evfimievski et al., 2002; Rizvi and Haritsa, 2002; Zhang et al., 2004), and classification (Zhang et al., 2005) are based on the altered data. However, the arbitrary randomization approach (Kargupta et al., 2003) is not fully secure to honest parties. Two data reconstruction methods (Huang et al., 2005), principal component analysis technique and Bayes estimate technique based on data correlations, can be used to solve the security issue. The distribution of the original data is normally reconstructed more accurately when correlation is higher in the randomization (Agrawal and Srikant, 2000; Agrawal and Aggarwal, 2001).

Secure Multi-party Computation (SMC) in PPDM. The second privacy-preserving technique in PPDM is based on SMC. In PPDM, we can apply SMC models such as Yao's circuit model (Section 2.5.1) and the homomorphic cryptography model (Section 2.5.3). Thus, current data mining algorithms need to be modified to allow privacy computation using SMC.

Comparison between the Randomization and SMC. In the PPDM algorithms, the randomization can achieve high efficiency in term of computation at the expense of information loss (i.e., obtaining less accurate results). In contrast, SMC can achieve the approximate complete accuracy, at the expense of expensive computation. Moreover, SMC can provide a complete privacy. In this thesis, our focus is to investigate SMC protocols that can achieve the approximate complete accuracy and provide a complete privacy yet efficient in term of computation. Thus, we will mainly investigate SMC in our thesis.

Privacy-preserving data mining uses SMC to compute functions over inputs provided by multiple parties without disclosing their inputs to each other. For simplicity, we consider PPDM in the two-party model (i.e., Alice and Bob). Let x and y be inputs of Alice and Bob respectively. Alice and Bob jointly compute the function $f(x, y)$ in which Alice learns nothing about y and Bob learns nothing about x. The two-party model can be

extended easily to support k parties, such that k parties can jointly to compute k arguments function $h(x_1, \cdots, x_k)$. In context of computation, many data mining algorithms are repetitive computations of many primitive functions (e.g., scalar dot product and division operation). To securely compute the functions $f(x, y)$ and $h(x_1, \cdots, x_k)$, any SMC protocol of PPDM must allow to transfer information in such a way that the functions are computed without violating privacy. In addition, the protocols are based on different adversarial models (Section 2.6): the semi-honest model and the malicious model. Many existing SMC protocols of PPDM use the semi-honest model. We will discuss more details in the following.

The security of the SMC protocols can be proven secure via simulation (Goldreich, 2004). Simulation is a standard methodology of proving the security of the SMC protocols in PPDM. We can simulate an SMC protocol of PPDM based on the semi-honest model (Section 2.6.1) as follows. Again, we use the case of two parties: Alice and Bob. Let π be a protocol of privately computing function f on (m_1, m_2), where m_1 and m_2 are the private inputs of Alice and Bob, respectively. Suppose that $f(m_1, m_2) = \{f_1(m_1, m_2), f_2(m_1, m_2)\}$, where $f_1(m_1, m_2)$ is the protocol output to Alice and $f_2(m_1, m_2)$ is that to Bob. Let $VIEW_1 = (m_1, V_1^1, V_1^2, \ldots, V_1^{t_1})$ be the view [1] of Alice, where V_1^i is the i-th message she receives from Bob. Let $VIEW_2 = (m_2, V_2^1, V_2^2, \ldots, V_2^{t_2})$ be the view of Bob, where V_2^i is the i-th message he receives from Alice.

The protocol π is simulatable, if 1) Alice can find a polynomial-time algorithm S_1 (i.e., a simulator), such that the distribution of $S_1(m_1, f_1(m_1, m_2))$ is indistinguishable from that of $VIEW_1$, and 2) Bob can find a polynomial-time algorithm S_2, such that the distribution of $S_2(m_2, f_2(m_1, m_2))$ is indistinguishable from that of $VIEW_2$. Intuitively, the privacy-preserving data mining protocol is simulatable, if neither Alice nor Bob learns additional information other than their respective inputs and the protocol outputs to them.

Many existing works of PPDM algorithms have been proposed to compute distributed data based on different data partitions (Figure 3.1). The distributed data can be partitioned in a horizontal partition or a vertical partition. In some cases (Jagannathan et al., 2006), the data can be split into an arbitrarily partition which may contain both of the horizontally and vertically partitioned data (i.e., each party holds different disjoint portions). (Jagannathan et al., 2006) argue that the arbitrarily partitioned data is not

[1] We assume honest-but-curious setting. Thus, the internal randomness of Alice (Bob) (Goldreich, 2004) in the view is uniform and can be ignored.

common in practice. They suggest that it is still good to use as a more general model of both horizontally and vertically partitioned data in practical settings. Therefore, in any of data partitions above, no any individual party holds the entire data for mining computation.

In PPDM, secure multi-party computation (SMC) can use the circuit approach and the homomorphic encryption protocol for secure computations in data mining tasks. In the circuit approach (Yao's circuit model in Section 2.5.1 and arithmetic circuit model in Section 2.5.2), SMC can apply the 1-out-of-2 oblivious transfer (OT) as a basic primitive of secure computation. Most solutions of the circuit approach in PPDM are based on the semi-honest model. Extending the 1-out-of-2 OT protocol to the-1-out-N OT protocol (or the k-out-of-N OT protocol) can be found in (Naor and Pinkas, 2001; Chaum et al., 1988; Yao, 1986). PPDM can also apply the circuit approach of SMC to compute some data mining primitives related to vector distances in the multi-dimensional space.

(Ioannidis et al., 2002; Du and Atallah, 2001a) propose a number of privacy-preserving data mining primitives, such as the scalar dot-product in a distributed environment, to address computational and communication overheads in the circuit approach of SMC. (Du and Atallah, 2001a) propose a framework to transform traditional data mining problems, such as classification, association rule mining, clustering, and generalization, into SMC problems. Many of these proposed techniques (Ioannidis et al., 2002; Du and Atallah, 2001a) send changed and encrypted inputs to participating parties so as to compute the function. The final output of the function is retrieved by an oblivious transfer (OT) protocol.

(Clifton et al., 2002) propose another set of methods based on the randomization and SMC for privacy-preserving data mining primitives. The methods of (Clifton et al., 2002) include secure set union (SMC), secure sum (the randomization), secure scalar product (the randomization) and secure size of set intersection (SMC). Those methods can securely compute data mining primitives on data which is partitioned vertically or horizontally. In SMC, Shamir's secret sharing (Shamir, 1979) and the semi-homomorphic encryption protocol (Section 2.1) also can be applied in computing functions in many data mining primitives. The homomorphic property of the semi-homomorphic encryptions (Section 2.1) allows to compute more complex functions in many data mining primitives. However, the fully-homomorphic encryptions (Section 2.2) can also provide the additive and

multiplicative homomorphic properties, but they are still impractical to apply into many data mining tasks. One of the several reasons (Yi et al., 2014) is that the computation performance of the fully-homomorphic encryption scheme is more underperformed than the semi-homomorphic encryption scheme.

In next section, we discuss secure building blocks in computing data mining primitives in PPDM. In Section 3.2, we discuss the state-of-the-art PPDM algorithms using the secure building blocks. The last section we discuss some other privacy preservation techniques.

3.1 Secure Building Blocks

In this section, we discuss secure building blocks of PPDM in detail. The secure building blocks can apply the randomization and SMC. The techniques of SMC in PPDM can include 1-out-of-2 oblivious transfer (OT), oblivious polynomial evaluation (OPE), the secret sharing, homomorphic encryption, and other cryptographic methods. All secure blocks of PPDM we discussed are based on the semi-honest model unless stated otherwise. We discuss 10 secure building blocks as follows: secure sum (Section 3.1.1), secure scalar product (Section 3.1.2), secure matrix multiplication (Section 3.1.3), secure set computation (Section 3.1.4), secure permutation (Section 3.1.5), oblivious polynomial evaluation (Section 3.1.6), secure logarithm (Section 3.1.7), secure division (Section 3.1.8), least significant bits gate (Section 3.1.9) and fast garbled circuit (Section 3.1.10).

3.1.1 Secure Sum

(Yao, 1986) introduces the general concept of secure multi-party computation (SMC). The basic idea of SMC is to allow multiple parties to securely compute a function over their inputs, while keeping respective inputs of the parties private. In other words, all the parties learn nothing except their own inputs and the computation result. Secure multi-party sum (SMS) performs the secure computation of the sum on the inputs of the multiple parties. Secure multi-party sum uses the randomization to compute the sum of data inputs.

The Randomization. Let 8, 12, -2, and 14 be private values held by parties P_1, P_2, P_3, and P_4 respectively, as depicted in the Figure 3.2. They jointly compute the sum of

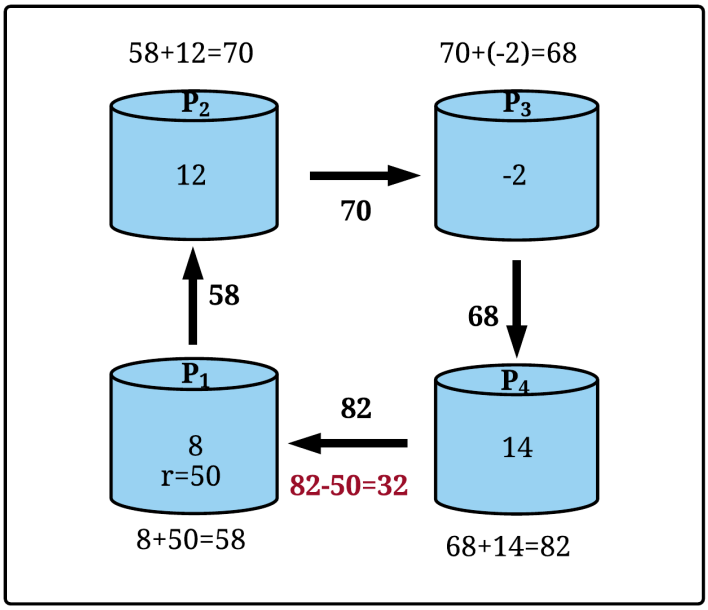


Figure 3.2: Secure computation of the sum of four parties

the input values without disclosing their own inputs by applying secure multi-party sum protocol. The steps for computing the sum of four parties (Figure 3.2) are in the following,

- P_1 first generates a random number, $r \in [-\infty, +\infty]$, and then adds r to its own private value. For simplicity, in Figure 3.2, r and the private input of P_1 are set to 50 and 8 respectively. The sum of r and the private input of P_1 is 58 which is sent to P_2. P_2 learns nothing about the private input of P_1 if r is sufficiently large.

- P_2 receives the value from P_1 and adds it to the private value of P_2. The sum is 70 which is sent to P_3. P_3 and P_4 repeat a similar step of P_2 to get 68 (i.e., $70 + (-2)$) and 82 (i.e., $68 + 14$), respectively.

- In the last step, P_1 receives the value 82 from P_4. P_1 subtracts the received value from the random value r, to get 32 (i.e., $82 - 50$) which is the result of the sum of the values of the parties. P_1 broadcasts the result 32 to other parties (i.e, P_2, P_3, and P_4). In the end of secure sum computation, all the parties learn only the sum of the values with their own inputs.

One of the obvious problems in secure sum is the private value of one party can be known if two neighboring parties collude with each other. In Figure 3.2, P_2 colludes with P_4 to reveal the private value of P_3. P_2 sends the sum of 70 to both P_3 and P_4. After

receiving the value 70, P_3 adds 70 to its own input -2 to get 68 (e.g., $70 + (-2) = 68$) which is sent to P_4. The private value of P_3 can be easily inferred by P_4 which subtracts the sum of P_2 from the sum of P_3 ($P_3 - P_2 = 68 - 70 = -2$).

Formal method (Kantarcioglu and Clifton, 2004) of secure sum computes a sum over private inputs of the different parties is discussed as follows. Let $S_1, S_2, \cdots, S_k$ be parties that involve the secure sum protocol. Each party S_i is given a private value v_i where $i \in 1, \cdots, k$. To compute $v = \sum_{i=1}^{k} v_i$ where $v \in 0, \cdots, n-1$ and $n \in \mathbb{Z}$, the parties apply secure sum. Party S_1 acts as the master site to initialize the secure sum protocol. S_1 first generates a random number r that is uniformly selected from $[0, \cdots, n-1]$. Then r is added to the value v_1 to get $r + v_1$ mod n which is sent to the next party S_2. Since $r + v_1$ mod n is uniformly distributed from $[0, \cdots, n-1]$, S_2 learns no information about v_1. The remaining of the parties, $\forall_{i=2}^{k-1} S_i$, perform secure sum computation as follows, (i) S_i receives the value $r + \sum_{j=1}^{i-1} v_j$ mod n from S_{i-1}. S_i learns no information about the received value which is uniformly distributed from $[0, \cdots, n-1]$. (ii) S_i adds its own input v_i to the received value so as to get the sum of $r + \sum_{j=1}^{i} v_j$ mod n. (iii) S_i sends the sum to the next party S_{i+1}.

Next, party S_k performs a similar sum computation as above to get $r + \sum_{i=1}^{k} v_i$ mod n $= r + v_1 + \sum_{i=2}^{k} v_i$ mod n which is sent to party S_1. After receiving the value, S_1 computes the sum of all values of the parties by subtracting r from the received value. S_1 can determine $\sum_{i=2}^{k} v_i$ by subtracting $r + v_1$. In the end of computation, S_1 learns nothing except the sum of all values of the parties. In a case, some parties can collude with each other to reveal values of an other party. For example, S_i can collude with S_{i+2} to reveal the private value of S_{i+1}. S_{i+2} receives the sum from S_i that can be combined with the sum from S_{i+1} to compute the value v_{i+1} of S_{i+1}. Secure sum is extended to work for an honest majority. Each party S_i splits v_i into a few shares. The sum is computed individually for each share. The path to compute each share is permuted to provide that each party has the same neighbor at most 1. Thus, to infer an other party value, e.g., v_i of S_i, the neighbors of S_i from each iteration requires to collude with each other. The number of splitting shares and that of dishonest parties can determine whether privacy is violated in the above protocol. The protocol is detailed in (Chor and Kushilevitz, 1993).

SMC. The secret scheme of Shamir (Shamir, 1979) is based on polynomial interpolation: having k points in the 2-dimensional plane $(x_1, y_1), (x_2, y_2), \cdots, (x_k, y_k)$ with distinct x's,

the scheme provides only one polynomial $q(x)$ of degree $k-1$ where $\forall_i^k q(x_i) = y_i$. In other words, the set of integers modulo p (i.e., a prime number) can form a field in which interpolation is possible in the scheme. For a simplicity, let D be a number. To divide it into a few pieces D_i, the scheme selects a random $k-1$ degree polynomial as $q(x) = a_0 + a_1 x + \cdots + a_{k-1}x^{k-1}$ where $a_0 = D$, to evaluate

$$D_1 = q(1), \cdots, D_i = q(i), \cdots D_n = q(n).$$

The scheme selects a prime number p that is bigger than both D and n. All integer coefficients $a_1, a_2, \cdots, a_{k-1}$ are selected randomly from the uniform distribution in $[0, p)$. All values $D_1, \cdots, D_n$ are computed in modulo p.

(Emekçi et al., 2007) apply the secret scheme of Shamir to perform secure sum of private inputs of the parties without revealing the inputs to each other . For example, there are four parties $(P_1 - P_4)$ jointly compute the summation of 4 party inputs without revealing their inputs respectively to each other. Let v_1, v_2, v_3 and v_4 be inputs of P_1, P_2, P_3 and P_4 respectively. They jointly compute $v_1 + v_2 + v_3 + v4$ and decide on a polynomial degree $k = 3$ and $m = 4$ values $X = 3, 5, 7, 8$. Each P_i then selects a random polynomial $q_i(x)$ of degree $k = 3$ which the constant term of $q_i(x)$ is the secret value v_i. The shares computed by each $P_i(X)$ are sent to other parties. Subsequently, each party combines all shares to get an immediate result which is sent to other party. In the last step, each party has 4 equations with 4 unknown coefficients (including the sum of the inputs of the parties). Thus, each party can solve the set of equations to determine the sum of the private inputs. The steps to perform secure sum using the secret scheme of Shamir is depicted in Protocol 2.

3.1.2 Secure Scalar Product

(Du and Zhan, 2002) propose a secure scalar product (SSP) protocol (the randomization) to compute the best split of records that contain a set of attribute values to construct a decision tree. The records are from vertically partitioned data. To evaluate each best split, they apply SSP to securely compute information gain and entropy. In the associate rule mining, (Zhan et al., 2007; Vaidya and Clifton, 2002) (the randomization) and (Zhong, 2007) (SMC) propose various secure scalar product protocols to compute confidence and

Protocol 2: Secure Sum using the Secret Scheme of Shamir

Input: Parties $P_1, P_2, \cdots, P_n$ have private values $v_1, v_2, \cdots, v_n$, respectively. Let $x_1, x_2, \cdots, x_n$ be a set of n known random values where $\forall_{i=1}^{n} x_i \neq 0$. Each random polynomial of degree k (i.e. $n-1$).

Output: $P_1, P_2, \cdots, P_n$ learns the sum of all values (i.e., $v_1 + v_2 + \cdots + v_n$).

1 Selects a random polynomial $q_i(x)$ of degree k, such that $q_i(x) = a_{k-1}x^{k-1} + \cdots + a_1 x^1 + v_i$.
2 Each party P_j computes the share, $sh(v_i, P_j) = q_i(x_j)$.
3 **for** $j \leftarrow 1$ **to** n **do**
4 | Send $sh(v_i, P_j)$ to peer P_j.
5 **end**
6 Each party P_i receives the shares $sh(v_j, P_i)$ from every party P_j.
7 Each party P_i computes an intermediate result, $\text{INTERRES}_i = \sum_{j=1}^{n} sh(v_j, P_i)$.
8 **for** $j \leftarrow 1$ **to** n **do**
9 | Send INTERRES_i to peer P_j.
10 **end**
11 Each party P_i receives the intermediate results INTERRES_j from every party P_j.
12 Each party P_i solves the set of equations to get the sum of $v_1 + v_2 + \cdots + v_n$ (i.e., $\sum_{j=1}^{n} v_j$).

support of an association rule. Both confidence and support can determine whether the rule is frequent.

In K-means clustering, (Jagannathan and Wright, 2005) apply SSP (SMC) to compute the closest cluster targeted on arbitrary partitioned data. (Vaidya and Clifton, 2004) also apply SSP (the randomization and SMC) to compute the estimated probability of each class label in Naïve Bayes classifier which data is split vertically. (Wright and Yang, 2004) also apply SSP (SMC) to learn the Bayesian network structure (using K2 algorithm) for vertically partitioned data. (Yu, Jiang and Vaidya, 2006) use SSP (SMC) to compute the Gram matrix for horizontally partitioned data. In the neural network, (Barni et al., 2006) apply SSP (SMC) to compute the weighted sum of input data by scalar product of the input and weight of the neurons. In summary, secure scalar product (SSP) is a fundamental block that is widely applying in many data mining algorithms in conjunction with the semi-honest model.

(Du and Atallah, 2001b) propose a secure scalar product product protocol based on the randomization and SMC. However, (Goethals et al., 2004) also propose a scalar product protocol (SMC) that is more efficient in computation and is proven secure as follows. Let

Protocol 3: Secure Scalar Product Protocol

Input: Alice has a vector $X = [x_1, x_2, \cdots, x_n]^T$ and Bob has a vector $Y = [y_1, y_2, \cdots, y_n]^T$.

Output: Alice gets r^A and Bob gets r^B where $r^A + r^B = X \cdot Y$.

1 Alice generates a pair of keys (sk, pk) (i.e., (secret key, public key)).
2 Alice sends pk to Bob.
3 **for** *i = 1 to n* **do**
4 | Alice encrypts $c_i = E_{pk}[x_i]$ which is sent to Bob.
5 **end**
6 Bob computes $\omega = \prod_{i=1}^{\pi} c_i^{y_i}$.
7 Bob generates a random number r^B to compute $\omega' = \omega \cdot E_{pk}[-r^B]$ which is sent to Alice.
8 Alice computes $r^A = D_{sk}[\omega'] = X \cdot Y - r$.

input x of Alice and input y of Bob be n-dimensional vectors. At the end of the SSP execution, Alice gets $r^A = x \cdot y - r^B$ and Bob gets r^B where r^B is a random number.

The main idea behind the SSP protocol (Goethals et al., 2004) is applying the semi-homomorphic encryption scheme to perform scalar dot product. Thus, many semi-homomorphic encryptions can be applied into the above SSP protocol such as Benaloch encryption (Benaloh, 1987), blum-Goldwasser encryption (Blum and Goldwasser, 1984), Naccahe-Stern encryption (Naccache and Stern, 1998), Okamoto-Uchiyama encryption (Okamoto and Uchiyama, 1998) and Paillier encryption (Paillier, 1999). The semi-homomorphic encryption schemes are proven semantically secure. Some semi-homomorphic encryption schemes are discussed in Section 2.1.

We describe the steps of the SSP protocol (Goethals et al., 2004). In the two-party setting, Alice and Bob can jointly compute scalar dot product as follows. The key idea is to compute $\sum_{i=1}^{n} x_i \cdot y_i = \sum_{i=1}^{n} (x_i + x_i + \cdots + x_i)(y_i \text{ times of } x_i)$. Alice first encrypts the vector $(x_1, x_2, \cdots, x_n)$ and sends it to Bob. After receiving the encrypted vector, Bob computes scalar dot product of the vector $(y_1, y_2, \cdots, y_n)$ with the encrypted vector of Alice using the semi-homomorphic encryption scheme. SSP (Goethals et al., 2004) is detailed in Protocol 3.

Malicious Model. (Kantarcioglu and Kardes, 2007) propose a secure scalar product protocol (SMC) which is against the malicious party using zero-knowledge proof. (Jiang and Clifton, 2007) propose the Accountable Computing (AC) framework to detect malicious behaviors by a third independent entity. The computation of AC approach is more efficient as it only initializes the identification and exposure of a malicious party (when an

honest party violates the protocol). They enhance the SSP protocol (SMC) by integrating it into the AC framework to withstand attacks from the malicious party.

3.1.3 Secure Matrix Multiplication

We can securely compute matrix multiplication using either the randomization or secure multi-party computation..

The Randomization. (Cramer and Damgård, 2001; Bar-Ilan and Beaver, 1989) show that matrix multiplication can be securely computed via a constant number of rounds of interaction among parties. One of the rounds is that each party sends one message to other participating parties. This technique has been proven secure in the information-theoretic sense. (Du et al., 2004) use the linear algebraic methods (Cramer and Damgård, 2001; Bar-Ilan and Beaver, 1989) to perform matrix multiplication. They use a random and invertible matrix M to hide the original matrix from privacy violation. Let M be $N \times N$ matrix such that

$$M = \begin{pmatrix} M_{left} & M_{right} \end{pmatrix}, M^{-1} = \begin{pmatrix} M_{top}^{-1} \\ M_{bottom}^{-1} \end{pmatrix}, \tag{3.1}$$

where $M \cdot M^{-1} = 1$. We next discuss the method of (Du et al., 2004) for secure matrix multiplication. Let A and B be matrices of Alice and Bob respectively. Alice and Bob jointly compute $A.B = R^A + R^B$ where R^A and R^B are held by Alice and Bob, respectively. The steps to compute $A.B$ are in the following.

i Alice and Bob jointly generate a random invertible matrix M $(N \times N)$.

ii Alice computes $A_1 = A \cdot M_{left}$ and $A_2 = A \cdot M_{right}$. She sends A_1 to Bob.

iii Bob computes $B_1 = M_{top}^{-1} \cdot B$ and $B_2 = M_{bottom}^{-1} \cdot B$. He sends B_2 to Alice.

iv Alice computes $R^A = A_2 \cdot B_2$, and Bob computes $R^B = A_1 \cdot B_1$.

Clearly, above the step (iv), $R^A + R^B$ is equal to

$$R^A + R^B = \begin{pmatrix} A_1 & A_2 \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} = AM \cdot M^{-1}B = A.B.$$

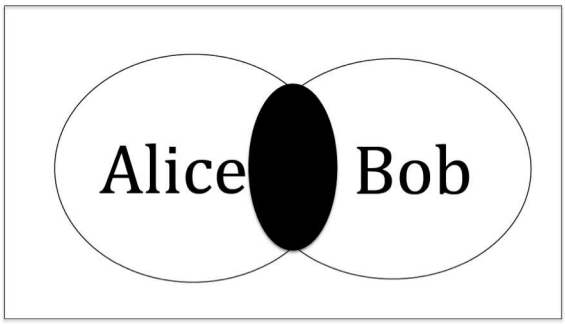


Figure 3.3: Set intersection between Alice and Bob.

In the above protocol, matrix A of Alice can be attacked by Bob or matrix B of Bob can be attacked by Alice. To avoid the attack, both Alice and Bob can generate a "k-secure" matrix M with the following conditions, such that (i) MA (resp. MB) includes at least $k+1$ unknown elements of A (resp. B), and (ii) at least $2k$ unknown elements of A (resp. B) are in any k combined equations. Many unknown elements on insufficient equations can allow infinitely possible solutions. Therefore, it is hard to know any element of matrix A (resp. B).

SMC. The secure matrix multiplication protocol (Du et al., 2004) can still be attacked if it runs many times with the same matrix A (resp. B); more equations (M's) are generated at the fixed unknown elements of matrix A (resp. B). Another issue is a complex process to construct the matrix M in the protocol. To address above limitations, (Han et al., 2010) propose a secure matrix multiplication by applying the SSP protocol of (Goethals et al., 2004).

3.1.4 Secure Set Computation

SMC. (Clifton et al., 2002) propose an SMC protocol to securely compute a set union based on commutative encryption for multiple parties. (Vaidya and Clifton, 2005) also propose an SMC protocol to securely compute a set intersection cardinality of multiple parties based on commutative one-way hash function (e.g., Pohlig-Hellman). (Agrawal et al., 2003) propose various secure set computation protocols such as equijoin, set intersection, intersection size, and equijoin size for the two-party model. (Agrawal et al., 2003) also formulate the notion of minimal information sharing across different private databases.

Protocol 4: Secure Set Intersection Protocol

Input: Alice has a dataset X and Bob has a dataset Y.

Output: Alice and Bob get $X \cap Y$.

1 Alice generates a secret key sk_a and Bob generates a secret key sk_b.
2 Alice encrypts her data with sk_a to get $V_a = E_{sk_a}[X]$, and Bob encrypts his data with sk_b to get $V_b = E_{sk_b}[Y]$.
3 Alice sends V_a to Bob, and Bob sends V_b to Alice.
4 Alice encrypts V_b with sk_a to get $W_a = E_{sk_a}[V_b]$, and Bob encrypts V_a with sk_b to get $W_b = E_{sk_b}[V_a]$.
5 Alice and Bob jointly find the intersection of W_a and W_b, and then decrypt the matching records.

Protocol 5: Secure Permutation Protocol

Input: Alice has an input vector $X = [x_1, x_2, \cdots, x_n]^T$ and Bob has an input random vector $R = [r_1, r_2, \cdots, r_n]^T$ with a permutation τ

Output: Alice gets $\tau(X + R)$

1 Alice generates a pair of keys (sk, pk). Alice keeps the secret key sk and sends the public key pk to Bob. Let $E[.]$ and $D[.]$ be encryption and decryption respectively.
2 Alice encrypts X to get $E[X] = ([E[x_1], E[x_2], \cdots, E[x_n]]^T)$ which is sent to Bob.
3 Bob computes $E[X] \cdot E[R] = E[X + R]$, and then uses the random function τ to permutate $E[X + R]$ (i.e., $\tau(E[X + R])$) which is sent to Alice.
4 Alice decrypts $\tau(E[X + R])$ to get $D[\tau(E[X + R])] = \tau(D[E[X + R]]) = \tau(X + R)$.

Secure set intersection (Agrawal et al., 2003) is to find the intersection of two different datasets that are given in the following. Let X and Y be datasets, held by Alice and Bob respectively. Alice and Bob can jointly find the intersection $X \cap Y$ of their datasets as depicted in Figure 3.3 Let $E_x[.]$ be an encryption function of Alice and $E_y[.]$ be an encryption function of Bob. Based on the property of the commutative encryption, the main idea of the protocol is that two data records are same if the encryptions of two data records are same ($E_y[E_x[X]] = E_x[E_y[Y]]$). Alice and Bob encrypt their datasets with $E_x[.]$ and $E_y[.]$, and then compare the encrypted data records. Finally, they jointly decrypt the matching records to get the result. Secure set intersection is depicted in Protocol 4. More analysis details can be found in (Agrawal et al., 2003).

3.1.5 Secure Permutation

SMC. (Du and Atallah, 2001b) propose a technique to compute $\tau(X + R)$ based on semi-homomorphic encryption (refer to Section 2.1 for more details). The semi-homomorphic encryption scheme has an additive property, such that $E[x] * E[y] = E[x + y]$ where $E[.]$ stands for encryption. Giving a vector $z = (z_1, z_2, \cdots, z_n)$, the encryption of z is $E[z] =$

$(E[z_1], E[z_2], \cdots, E[z_n])$, and the decryption of z is $D[z] = (D[z_1], D[z_2], \cdots, D[z_n])$ where $D[.]$ stands for decryption. Let X be a vector of Alice, and τ and R be a random function and a random vector respectively, that of Bob. The method of (Du and Atallah, 2001b) computes $\tau(X + R)$ that is detailed in Protocol 5.

Theorem 3.1 *The permutation algorithm of the Protocol 5 can privately compute a permuted vector sum of two party vectors, where Alice learns the permuted sum $\tau(X+R)$ and Bob learns the permutation τ.*

(Du and Atallah, 2001b) prove the above protocol via a simulation. In the simulation, view of Bob is created as follows. Bob receives a encrypted vector $E[X]$ of length n from Alice. Bob selects a random vector R' to encrypt with the public key pk of Alice. Since the encryption scheme is semantically secure, $E[X]$ and $E[R']$ are computationally indistinguishable.

Next, the view of Alice is created as follows. Alice receives a vector $\tau(E[X + R])$ with a size of n from Bob. Alice generates a vector of random numbers with a size of n and then encrypts the random vector. Since the encryption scheme is semantically secure, the received vector and the random vector are computationally indistinguishable. Alice and Bob each learn no information other than respective inputs and the protocol outputs to them (if any). Thus, the secure permutation protocol is secure.

3.1.6 Oblivious Polynomial Evaluation (OPE)

SMC. (Naor and Pinkas, 1999) propose an oblivious polynomial evaluation (OPE) using the oblivious transfer protocol (refer to Section 2.3 for more details). The oblivious polynomial evaluation involves a sender and a receiver. Let sender be Alice and receiver be Bob. The input of Alice (i.e., the sender) is a polynomial Q of degree d_p which is defined over some finite field $\mathcal{F}$. The parameter d_p is public. The input of Bob (i.e., the receiver) is an element $z \in \mathcal{F}$. At the end of the execution of the OPE protocol, Bob learns $Q(z)$ without learning anything about the polynomial Q, and Alice learns nothing. Thus, the functionality of the oblivious polynomial evaluation can be defined as: $(Q, z) \mapsto (\lambda, Q(z))$. The steps to perform oblivious polynomial evaluation are detailed in Protocol 6. The OPE protocol uses $d + d_p + 1$ coefficients to define the polynomial Q. The overhead of the protocol is from the line 3 in Protocol 6 that involves interaction between Alice and Bob.

Protocol 6: Oblivious Polynomial Evaluation

Input: Alice (sender) defines a polynomial $P(y) = \sum_{i=0}^{d_p} b_i y^i$ of degree d_p in the field $\mathcal{F}$. For a simplicity, let y be an input to P. Bob (receiver) has a value $\alpha \in \mathcal{F}$.

Output: Bob learns $P(\alpha)$.

1 Alice uses a bivariate polynomial to hide P: She generates a random masking polynomial $P_x(x)$ of degree d, such that $P_x(0) = 0$ where $P_x(x) = \sum_{i=1}^{d} a_i x^i$. The parameter d equals to the security parameter k multiplied by the degree of P (i.e., $d = k \cdot d_p$), where k is a security parameter. The bivariate polynomial is defined as follows,

$$Q(x, y) = P_x(x) + P(y) = \sum_{i=1}^{d} a_i x^i + \sum_{i=0}^{d_p} b_i y^i,$$

where $\forall_y Q(0, y) = P(y)$.

2 Bob uses a univariate polynomial to hide α: He selects a random polynomial S of degree k where $S(0) = \alpha$. She uses the univariate polynomial $R(x) = Q(x, S(x))$ to learn $P(\alpha)$ from Alice.

3 Bob learns the points of R: He learns $d_R + 1$ values in the form of $\langle x_i, R(x_i)\rangle$.

4 Bob computes $P\alpha$: He uses the values of R to interpolate $R(0) = P(\alpha)$. Since the following condition $R(0) = Q(0, S(0)) = P(S(0)) = P(\alpha)$ holds, Bob can interpolate R to learn $R(0) = P(\alpha)$ (where the degree of R is $d_R = d = k \cdot d_p$).

(Naor and Pinkas, 2001) improve the computation performance of the OPE protocol using one-out-of-N oblivious transfer. More details of the OPE protocol can be found in (Naor and Pinkas, 2001, 1999). Alternatively, any homomorphic encryption scheme (refer to Section 2.1 for more details) also can be used to implement an OPE protocol.

3.1.7 Secure Logarithm

SMC. (Lindell and Pinkas, 2002, 2000) propose an SMC protocol to compute $\ln x$ based on Yao's protocol (Section 2.5), the oblivious transfer protocol (Section 2.3), and the oblivious polynomial evaluation (OPE) protocol (Section 3.1.6). In the natural logarithm, $\ln(x + \epsilon)$ can be expanded by Taylor series as follows,

$$\ln(1 + \epsilon) = \sum_{i=1}^{\infty} \frac{-1^{i-1}\epsilon^i}{i} = \epsilon - \frac{\epsilon^2}{2} + \frac{\epsilon^3}{3} - \frac{\epsilon^4}{4} + \cdots, \tag{3.2}$$

where $-1 < \epsilon < 1$. In above series, it is easy to get the error for a partial evaluation in the following.

$$\left| \ln(1 + \epsilon) - \sum_{i=1}^{k} \frac{(-1)^{i-1}\epsilon^i}{i} \right| < \frac{|\epsilon|^{k+1}}{k+1} \cdot \frac{1}{1 - |\epsilon|}. \tag{3.3}$$

Protocol 7: Secure Logarithm

Input: Alice has an input v^A and Bob has an input v^B where $v^A + v^B = x = 2^n(1+\epsilon)$, 2^n is nearest to x, and $-1 < \epsilon < 1$.

Output: Alice gets u^A and Bob gets u^B where $u^A + u^B = \text{lcm}(2, \cdots, k) \cdot 2^N \ln(v^A + v^B)$.

1 Alice and Bob jointly run Yao's protocol on inputs v_A and v_B to compute (i) $\epsilon 2^N \mod |\mathcal{F}|$ to get α^A and α^B, held by Alice and Bob respectively, and (ii) $2^N \cdot n\ln(2) \mod |\mathcal{F}|$ to get β^A and β^B, held by Alice and Bob respectively.

2 Alice selects $z_1 \in_R \mathcal{F}$ to define the polynomial, $Q(z) = \text{lcm}(2, \cdots, k) \cdot \sum_{i=1}^{k} \frac{(-1)^{i-1}}{2^{N(i-1)}} \frac{(\alpha^A + z)^i}{i} - z_1$.

3 Alice and Bob jointly execute a (private) polynomial evaluation with Alice inputting $Q(.)$ and Bob inputting α^B. Bob gets $z_2 = Q(\alpha^B)$.

4 Alice sets $u^A = \text{lcm}(2, \cdots, k)\beta^A + z_1$ and Bob sets $u^B = \text{lcm}(2, \cdots, k)\beta^B + z_2$.

Obviously, the error in Equation 3.3 shrinks exponentially as k grows. Thus, $\ln(x)$ can be expressed into

$$\ln(x) \approx \ln(2^n(1+\epsilon)) = n\ln(2) + \epsilon - \frac{\epsilon^2}{2} + \frac{\epsilon^3}{3} - \frac{\epsilon^4}{4} \cdots, \tag{3.4}$$

where 2^n is nearest to x. Subsequently, Equation 3.4 can be transformed into

$$\text{lcm}(2, \cdots, k) \cdot 2^N \left(n\ln(2) + \epsilon - \frac{\epsilon^2}{2} + \frac{\epsilon^3}{3} - \cdots \frac{\epsilon^k}{k} \right) \approx \text{lcm}(2, \cdots, k) \cdot 2^N \cdot \ln(x), \tag{3.5}$$

where lcm is the least common multiple of $(2, \cdots, k)$.

(Lindell and Pinkas, 2002, 2000) use Yao's protocol to compute the first element $2^N \cdot n\ln 2$ in the series of $\ln(x)$ (Equation 3.5) and the remainder of the series $\epsilon \cdot 2^N$ (Equation 3.5) where N is a public upper-bound of the value n ($N > n$). The next step is to define and evaluate a polynomial. We use the two-party model (e.g., Alice and Bob) in the protocol. Alice first defines a polynomial $Q(z)$. Alice and Bob then compute $Q(z)$ using oblivious polynomial evaluation to get z_1 and z_2, held by Alice and Bob respectively, where $z_1 + z_2 = \text{lcm}(2, \cdots, k) \cdot 2^N \cdot \ln(x)$ (Equation 3.5). The steps are detailed in Protocol 7. More details of the secure logarithm protocol can be found in (Lindell and Pinkas, 2002, 2000).

(Ryger et al., 2008) propose a few approaches to optimize secure logarithm of (Lindell and Pinkas, 2002, 2000). To make integer $(\epsilon \cdot 2^N)^i$ is divisible by the integer $2^{N(i-1)}$, they use 2^{Nk} to replace 2^N in Equation 3.5. They also suggest to use $2^{Nk+2k+\log N}$ that can

support larger protocol in computing $x\ln(x)$. To support non-integer in secure logarithm, the solutions of (Ryger et al., 2008; Lindell and Pinkas, 2002, 2000) are not discussed in detail. The precision of secure logarithm is bounded by the number of iterations k in the polynomial Q (line 2 in Algorithm 7). Since the oblivious polynomial evaluation (OPE) protocol is expensive in computation, the time complexity increases significantly as k increases in the polynomial Q. The reasons above have motivated us to investigate a new secure logarithm that can address above issues. Our new secure logarithm will be efficient in computation and effective in protecting data privacy.

3.1.8 Secure Division

SMC. (Bunn and Ostrovsky, 2007) propose a secure division that involves two parties (e.g., Alice and Bob). Let P and D be integers (i.e. $P, Q \in \mathcal{Z}_N$). Alice and Bob select a random value $R \in \mathcal{Z}_N$ (uniformly distributed), and $Q \in \mathcal{Z}_N$ remains secret to both Alice and Bob. Let P^A and P^B be inputs of Alice and Bob respectively, and D^A and D^B be inputs of Alice and Bob respectively, where $P^A + P^B = P$ and $D^A + D^B = D$. They apply secure division to compute the quotient Q of $\frac{P}{D}$ where $Q < N$, $0 \leq R < D$, and $P = QD + R$. At the end of the computation, the outputs are Q^A and Q^B, held by Alice and Bob respectively, where $Q^A + Q^B = Q$. Note that Q (an actual quotient in $\mathcal{R}$) has been rounded down to the closest integer. The secure division protocol of (Bunn and Ostrovsky, 2007) involves sub-protocols such as the γ protocol (Bunn and Ostrovsky, 2007) and the find minimum of 2 numbers protocol (Bunn and Ostrovsky, 2007). The two sub-protocol involves the secure scalar product protocol (refer to Section 3.1.2 for more detail). More details of this protocol can be found in (Bunn and Ostrovsky, 2007). The secure division protocol of (Bunn and Ostrovsky, 2007) can provide a complete privacy under the semi-honest model. However, the protocol has not yet been proven to be efficient, and its approximation error is not clear either.

(Dahl et al., 2012) use two steps to compute secure division, $\frac{n}{d}$. Let n and d be ℓ-bit integers, and k be a large integer. The steps are as follows: (i)Alice and Bob jointly compute an encrypted approximation of $[\tilde{a}]$ of $a = \lfloor \frac{2^k}{d} \rfloor$, (ii) Alice and Bob compute $[\lfloor \frac{n}{d} \rfloor]$ which is equal to $\lfloor \frac{([\tilde{a}] \cdot [n])}{2^k} \rfloor$.

In the first step, (Dahl et al., 2012) use a Taylor series to compute a "k-shifted" approximation of $\frac{1}{d}$ where d is an integer as follows,

$$\frac{1}{\alpha} = \sum_{i=0}^{\infty}(1-\alpha)^i = \sum_{i=0}^{\omega}(1-\alpha)^i + \epsilon_\omega, \tag{3.6}$$

where $\epsilon_\omega = \sum_{i=\omega+1}^{\infty}(1-\alpha)^i$. The approximation approach is similar in (Hesse et al., 2002) that uses the constant depth division circuit and in (Kiltz et al., 2005). The error ϵ_ω of Equation 3.6 is bounded by

$$\epsilon_\omega = \sum_{i=\omega+1}^{\infty}(1-\alpha)^i \leq 2^{-\omega-1}\cdot\frac{1}{\alpha} \leq 2^{-w}, \tag{3.7}$$

where $0 < 1-\alpha \leq \frac{1}{2}$. The error is relatively small by setting ω sufficiently large. Thus, ϵ_ω can be truncated in above computation.

In Equation 3.6, $\frac{1}{\alpha}$ is multiplied by a power of two "shifts" to ensure that each of $\omega+1$ terms is an integer. Let $\ell_d = \lfloor \log(d)+1 \rfloor$ be a bit length of d (i.e., $2^{\ell_d-1} \leq d < 2^\ell$). For giving $\alpha = \frac{d}{2^{\ell_d}}$ and $k = \ell^2 + \ell$, $\frac{2^k}{d}$ shifted up by k bits is computed as

$$\begin{aligned}
\frac{2^k}{d} &= 2^{k-\ell_d}\cdot\frac{1}{d/2^{\ell_d}} \\
&= 2^{k-\ell_d}\cdot\left(\sum_{i=0}^{\omega}\left(1-\frac{d}{2^{\ell_d}}\right)^i + \epsilon_w\right) \\
&= 2^{k-\ell_d(\omega+1)}\cdot\sum_{i=0}^{\omega}\left(1-\frac{d}{2^{\ell_d}}\right)^i\cdot 2^{\ell_d\omega} + 2^{k-\ell_d}\epsilon_w \\
&= 2^{k-\ell_d(\omega+1)}\cdot\sum_{i=0}^{\omega}\left(2^{\ell_d}-d\right)^i\cdot 2^{\ell_d(\omega-i)} + 2^{k-\ell_d}\epsilon_w.
\end{aligned} \tag{3.8}$$

Therefore, the approximation of $\tilde{a}$ is $2^{k-\ell_d(\omega+1)}\cdot\sum_{i=0}^{\omega}\left(2^{\ell_d}-d\right)^i\cdot 2^{\ell_d(\omega-i)}$.

The second step of (Dahl et al., 2012) is multiplied $[\tilde{a}]$ by $\frac{[n]}{2^k}$ which is the approximation of $\lfloor\frac{n}{d}\rfloor$. (Dahl et al., 2012) use many sub-protocols such as the greater-than protocol (Damgård et al., 2006), the inverse of the element protocol (Bar-Ilan and Beaver, 1989), bit-decomposition (Damgård et al., 2006), and the prefix-or of a sequence of bits protocol (Damgård et al., 2006), to perform secure division. More details of this protocol can be found in (Dahl et al., 2012). However, the secure division protocol of (Dahl et al., 2012) applies the expensive bit-decomposition to compute the bit length ℓ of d in the secure division. A total complexity of the bit-decomposition (Damgård et al., 2006) requires 114

rounds and $110\ell \log \ell + 118\ell$ invocations of secure multiplication (e.g., in the two-party model, each secure multiplication requires 9 modulus exponentiations (Cramer et al., 2001) to compute $E[a] \times E[b]$, held by Alice and Bob respectively, where $d = a + b$. Moreover, the solution has not been given any empirical results (Dahl et al., 2012).

In other secure division protocols, (Veugen, 2014) consider to use a public divisor or a private divisor (known by one of two parties) to perform secure division. (Su et al., 2007; Jha et al., 2005; Vaidya and Clifton, 2003) propose some methods that allow both parties to compute division on their locally data. (Jagannathan and Wright, 2005) perform secure division by mutliplication by inverse(Jagannathan and Wright, 2005) that is not the correct division operation(Bunn and Ostrovsky, 2007). All of the secure divisions (Veugen, 2014; Su et al., 2007; Jha et al., 2005; Jagannathan and Wright, 2005; Vaidya and Clifton, 2003) can not provide complete privacy under the semi-honest model (Bunn and Ostrovsky, 2007).

All secure division protocols we discussed are either not efficient in computation or not effective in protecting data privacy, and both of them in privacy-preserving data mining (PPDM). As the secure division protocol is important in many data mining tasks, this motivates us to investigate a new secure division that can address above issues. We will propose the secure division protocol that will be efficient in computation and also effective in protecting data privacy in PPDM.

3.1.9 Least Significant Bits Gate (LSBs)

SMC. The LSBs Gate (Schoenmakers and Tuyls, 2006) extracts the ℓ least significant encrypted bits of the plaintext m that is encrypted based on the threshold Paillier encryption (Cramer et al., 2001). Let use a $(2, 2)$-threshold Paillier encryption (i.e. generate two privates key, sk_{alice} and sk_{bob} and hence need both sk_{alice} and sk_{bob} to decrypt the ciphertext). The basis idea of LSBs Gate is both Alice and Bob jointly generate a random value r and compute $M = E[pk, m] \cdot E[pk, r_{alice}] \cdot E[pk, r_{bob}]$. Then they decrypt $y = D[sk_{alice}, M] + D[sk_{bob}, M]$ (i.e. $y = m + r$). The encrypted bits of $E[m_0, m_1], \cdots, E[m_{\ell-1}]$ of the plaintext m can be recovered from $y_0, y_1, \cdots, y_{\ell-1}$ and the encrypted bits of $E[r_0], [r_1], \cdots, E[r_{\ell-1}]$. For security, $r = \sum_{j=0}^{\ell-1} r_j 2^j + 2r*$ is a sufficiently large number where $r_0, \cdots, r_{\ell-1}$ are bits and $2*$ is a large integer. Therefore, LSBs Gate is a semantic secure in which $y = x - r$ is statistically indistinguishable from a random.

Protocol 8: Least Significant Bits Gate

Input: An encrypted message, $E[x]$, where $0 \leq x < 2^m$ and $m + k + \log n < \log N$ (Note that $m = \ell$ is in this protocol).

Output: An enrypted bits, $E[m]$

1 Alice and Bob jointly generate random bits $E[r_0], \cdots, E[r_{m-1}]$ using m random-bit gate (Schoenmakers and Tuyls, 2006). In parallel, Alice and Bob select $r_{,1}$ and $r_{*,2}$, respectively, where $\forall_{i=1}^{2} r_{,i} \in_R \{0, \cdots, 2^{m+k-1} - 1\}$. The encryption of $E[r]$ that is equal to $r = \sum_{i=1}^{2} r_{,i}$ is publicly computed.

2 Alice and Bob compute $E[x - r]$ and then jointly decrypt $E[x - r]$ to get the signed value $y = x - r \in (-\frac{n}{2}, \frac{n}{2})$, where $r = \sum_{j=0}^{m-1} r_j 2^j + r2^m$. The signed value y is computed in modulo n (i.e., $y \equiv x - r \bmod n$).

3 Let $y_0, y_1, \cdots, y_{m-1}$ be a binary representation of $y \bmod 2^m$. To get an output of m encrypted bits, the addition circuit (Schoenmakers and Tuyls, 2006) uses the inputs of $y_0, y_1, \cdots, y_{m-1}$ (public) and $E[r_0], E[r_1], \cdots, E[r_{m-1}]$.

The least significant bits (LSBs) gate involves a few sub-protocols such as the random gate (Schoenmakers and Tuyls, 2006) and the addition circuit (Schoenmakers and Tuyls, 2006) to extract ℓ bits of the encrypted message. The steps to perform the least significant bits (LSBs) gate are depicted in Protocol 8. LSBs gate can involve with more than two parties. In a case of $\ell < m$, (Schoenmakers and Tuyls, 2006) propose a technique to combine with the Protocol 8. More details of LSB's gate can be found in (Schoenmakers and Tuyls, 2006) (Note that the authors also provide a solution to extract the least significant bit instead of ℓ bits).

3.1.10 Fast Garbled Circuit

SMC. (Huang, Evans, Katz and Malka, 2011) propose the fast garbled circuit that consists of simple circuits such as the AND-gate , the OR-gate and the XOR-gate. By a combination of different gates, some secure computations (Huang, Evans, Katz and Malka, 2011; Huang, Malka, Evans and Katz, 2011) such as integer comparison, Hamming distance, and many more can be performed. The fast garbled circuit use the circuit approach (Sections 2.5.1 and 2.5.2) and oblivious transfer (Section 2.3) to perform secure computation. Numeric comparison is one of most frequent computations in many privacy-preserving data mining algorithms. In this thesis, integer comparison circuit which is part of the fast garbled circuit is given a name, CMP.

The CMP integrates the comparison circuit of (Kolesnikov et al., 2009) and combines the efficient oblivious transfer protocols of (Ishai et al., 2003) and (Naor and Pinkas, 2001) applying an aggressive pre-computation technique to perform secure integer comparison. Let x and y be the inputs of Alice and Bob respectively. The CMP compares two ℓ-bit integers x^ℓ (Figure 3.4(i)) and y^ℓ to get

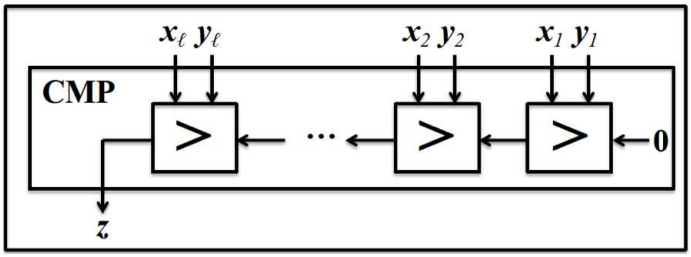
$$z = \begin{cases} 1 & \text{if } x^\ell > y^\ell, \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

The CMP forms Boolean circuits (i.e., ">") to evaluate $x^\ell > y^\ell$ using one 2-input AND gate with 4 table entries and three free XOR gates, as depicted in Figure 3.4(ii). Other comparisons like $x^\ell < y^\ell$, $x^\ell \geq y^\ell$, or $x^\ell \leq y^\ell$ are possible in the CMP that can switch x^ℓ with y^ℓ and/or setting the initial carry bit $c_1 = 1$ in the circuit (Figure 3.4(ii)).

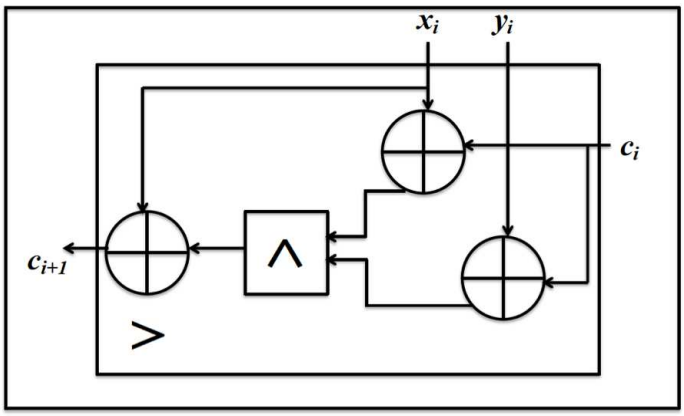
In the CMP, the oblivious transfer protocols involve two phases: (i) in preparation phase, two parties jointly pre-compute oblivious transfer, and then create and transfer the garble circuit via oblivious transfer. The preparation phase is a one-off initialization. (ii) in online phase, the circuit is evaluated so that it involves symmetric encryption (e.g., SHA-1) without any modular exponentiation. The online phase requires $2(k_1 + m)$ encryptions and $k_1 + m$ decryptions where k_1 is a security parameter (e.g., 80) with m pairs of ℓ-bit strings. Since the expensive modulus exponentiation is shifted to the preparation phase, CMP is the efficient protocol to securely compare integers between two parties.

3.2 Privacy-Preserving Data Mining Algorithms

In this section, we discuss some state-of-the-art privacy-preserving data mining (PPDM) algorithms that use some of secure building blocks as we discussed in Section 3.1. The PPDM algorithms have been proposed to mine the distributed data and to protect data privacy which data can be split vertically or horizontally, and both of them among participating parties. We first discuss privacy-preserving Naïve Bayes classifier in Section 3.2.1. Privacy-preserving support vector machine and privacy-preserving decision tree are discussed in Sections 3.2.2 and 3.2.3, respectively. We also discuss privacy-preserving association rule mining (Section 3.2.4), privacy-preserving clustering (Section 3.2.5) and other privacy-preserving data mining algorithms (Section 3.2.6). In Section 3.2.7, we discuss



(i) Comparison Circuit (CMP)



(ii) Comparator (">")

Figure 3.4: Comparison in the Fast Garbled Circuit

the limitations of the current PPDM algorithms and propose a general privacy model to address the limitations in PPDM.

3.2.1 Privacy-Preserving Naïve Bayes Classifier

A Bayesian classifier is a statistical classifier based on Bayes theorem. It can predict probabilities of class members; e.g., given a sample, the Bayesian classifier can calculate the probability of the sample that belongs to a particular class. To reduce complexity for learning the Bayesian classifier, the Naïve Bayes classifier assumes all attributes (i.e., features) are conditionally independent. (Domingos and Pazzani, 1996) show that the Naïve Bayesian learning is sufficiently effective in comparable to performance with other classifiers such as neural network and decision tree.

Let $A_1, A_2, \cdots, A_n$ be attributes that are conditionally independent with each other, given C. Thus, based on Bayes theorem, we can write it as

$$\begin{aligned} P(A|C) =& P(A_1, A_2, \cdots, A_n|C) \\ =& P(A_1|C)P(A_2|C)\cdots P(A_n|C) = \prod_{i=1}^{n} P(A_i|C). \end{aligned} \tag{3.10}$$

Next, we assume that (in general) C is a discrete variable and $A_1, A_2, \cdots, A_n$ are discrete or real attributes. Let $m_1, m_2, \cdots, m_k$ be values of C. Given a new instance A, the Naïve Bayes classifier can compute the probability C taking $m_i \in 1, \cdots, k$ as follows,

$$P(C = m_i|A_1, A_2, \cdots, A_n) = \frac{P(C = m_i)P(A_1, A_2, \cdots, A_n|C = m_i)}{\sum_{j=1}^{k} P(C = m_j)P(A_1, A_2, \cdots, A_n|C = m_j)}, \tag{3.11}$$

where the sum is added by each probability of all possible values of C. If $A_1, A_2, \cdots, A_n$ are conditional independent given C, we can substitute Equation 3.10 into 3.11 as

$$P(C = m_i|A_1, A_2, \cdots, A_n) = \frac{P(C = m_i)\prod_{i=1}^{n} P(A_i|C = m_i)}{\sum_{j=1}^{k} P(C = m_j)\prod_{i=1}^{n} P(A_i|C = m_j)}. \tag{3.12}$$

Thus, the probability C that takes any value can be computed as the observed attribute values of a new instance and the distributions $P(C)$ and $P(A_i|C)$ estimated from training data are given. Most probable value of C can find by

$$C \leftarrow \arg\max_{m_i} \frac{P(C = m_i)\prod_{i=1}^{n} P(A_i|C = m_i)}{\sum_{j=1}^{k} P(C = m_j)\prod_{i=1}^{n} P(A_i|C = m_j)}, \tag{3.13}$$

which can simplify to

$$C \leftarrow \arg\max_{m_i} P(C = m_i)\prod_{i=1}^{n} P(A_i|C = m_i). \tag{3.14}$$

More details of Naïve Bayes can be found in (Mitchell, 1997).

In the horizontally partitioned data, (Kantarcioglu and Clifton, 2003) propose privacy-preserving Naïve Bayes classifier based on the secure sum protocol (Section 3.1.1). (Vaidya, Kantarcioglu and Clifton, 2008; Vaidya and Clifton, 2004) use the secure scalar product protocol (Section 3.1.2) in privacy-preserving Naïve Bayes classifier targeted on vertically partitioned data. The secure scalar product protocol can compute the approximate probability without revealing input data to each other. (Wright and Yang, 2004) propose a privacy-preserving Naïve Bayesian network computation that involves with two parties

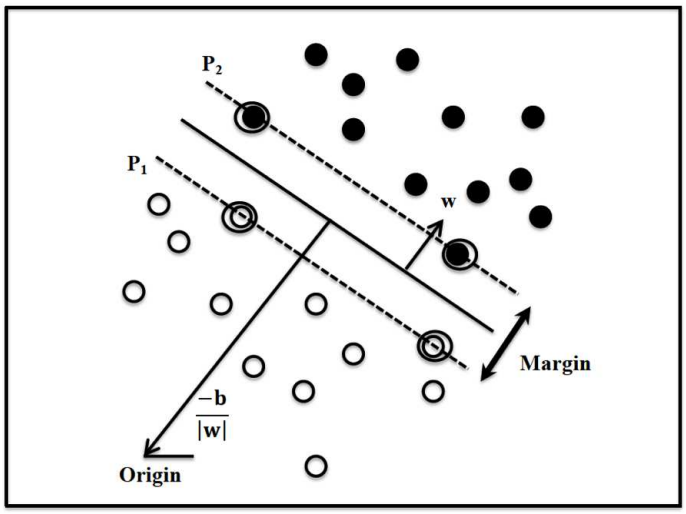


Figure 3.5: Linear separating hyperplanes by SVM. The support vectors are circled.

targeted on vertically partitioned data. The method of (Wright and Yang, 2004) is to enhance the $K2$ algorithm with privacy-preserving enabled that it can securely and efficiently construct the structure of a Bayesian network from input data of the two parties.

3.2.2 Privacy-Preserving Support Vector Machine

Support vector machine (SVM) is one of the popular data mining algorithms to perform tasks such as classification and regression. The objectives of SVM are to (i) maximize the geometrical margin of hyperplane separation and (ii) minimize the experimental classification error. In SVM, a maximal separating hyperplane is constructed by mapping input vectors into a higher dimensional space. Another two hyperplanes that are parallel to the separating hyperplane are constructed to separate data. Therefore, the separating hyperplane can either maximize or minimize the distance between the two parallel hyperplanes. Figure 3.5 shows a linear SVM that separates data between the hyperplanes. More details of SVM can be found in (Burges, 1998).

In the linear SVM (Figure 3.5), training data are linearly separated by optimal hyperplanes. Let x_i and y_i be data and the label of data respectively, where $x_i \in \mathbb{R}^d$, $i \in 1, \cdots, \ell$, and $y_i \in \{-1, +1\}$. Assuming that is a separating hyperplane can separate the negative ($y = -1$) from the positive ($y = +1$) label of data. Each data x_i which lies on the separating hyperplane satisfies $w \cdot x + b = 0$, where w is normal to the hyperplane, $\frac{|b|}{||w||}$ is the perpendicular distance from the origin to the hyperplane, and $||w||$ is the Euclidean form of w. The margin of the hyperplane is d_- (d_+) which is the shortest distance from

the hyperplane to the nearest negative (positive) label of data. To find the separating hyperplane with largest margin, all the training data need to satisfy the constraints as follows,

$$x_i \cdot w + b \leq -1 \qquad \text{for } y_i = -1, \tag{3.15}$$

$$x_i \cdot w + b \geq +1 \qquad \text{for } y_i = +1. \tag{3.16}$$

Equations 3.15 and 3.16 can be combined together to

$$\forall_i \quad y_i(x_i \cdot w + b) - 1 \geq 0. \tag{3.17}$$

Next consider two cases based on Equations 3.15 and 3.16. Assuming that x_i lies on the hyperplane $P_1 : x_i \cdot 2 + b = -1$ with normal w that is a perpendicular distance from the origin $\frac{|-1-b|}{||w||}$ (Equation 3.15). Similarly, x_i lies on the hyperplane $P_2 : x_i \cdot 2 + b = 1$ with normal w that is a perpendicular distance from the origin $\frac{|1-b|}{||w||}$ (Equation 3.16). Thus, $d_{\_} = d_+ = \frac{1}{||2||}$ is the margin to $\frac{2}{||w||}$ (Note that no training data fall between parallel hyperplanes P_1 and P_2). To find a pair of hyperplanes that gives the maximum margin, $||w||$ can be minimized using the Equation 3.17.

The constraint equations that are multiplied by Lagrange multiplier and subtracted from the objective function can be transformed into the Lagrangian form as

$$L_P \equiv \frac{1}{2}||w||^2 - \sum_{i=1}^{l} \alpha_i y_i (x_i \cdot w + b) + \sum_{i=1}^{l} \alpha_i, \tag{3.18}$$

where each of the inequality constraints in Equation 3.17 is multiplied by positive Lagrange multipliers $\alpha_i, i \in 1, \cdots, l$. To solve the dual problem in Equation 3.18, the gradient of L_P (i.e., the primal problem) with respect to w and b gives

$$w = \sum_{i=1} \alpha_i y_i x_i, \quad \sum_i \alpha_i y_i = 0.$$

Substituting above two conditions into Equation 3.18 can get L_D (i.e., the dual problem) as

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j. \tag{3.19}$$

Day	Outlook	Humidity	Wind	PlayBall
01	Sunny	High	Weak	No
02	Sunny	High	Strong	No
03	Rain	High	Weak	Yes
04	Rain	Normal	Weak	Yes
05	Rain	Normal	Strong	No

(a) Original dataset, S

Day	Outlook	Humidity	Wind	PlayBall
01	Sunny	-	Weak	No
02	-	High	-	No
03	Rain	-	-	Yes
04	Rain	-	-	Yes
05	-	Normal	Strong	No

(b) Alice's dataset

Day	Outlook	Humidity	Wind	PlayBall
01	-	High	-	No
02	Sunny	-	Strong	No
03	-	High	Weak	Yes
04	-	Normal	Weak	Yes
05	Rain	-	-	No

(c) Bob's dataset

Figure 3.6: Datasets of Alice and Bob. The class attribute is "PlayBall".

Above discussion is the linear SVM to find the optimal hyperplane. (Boser et al., 1992) propose a non-linear SVM using kernel functions to contruct classifiers. The non-linear SVM works in a simlar way as in the linear SVM with one exception. The exception is that every dot product of the linear SVM is changed to a kernel function that fits the maximum-margin hyperplane in the feature space.

(Yu, Jiang and Vaidya, 2006) use the secure intersection cardinality protocol (Vaidya and Clifton, 2005) (Section 3.1.4) to construct the global SVM classification model by kernel function targeted on horizontally partitioned data. (Yu, Vaidya and Jiang, 2006) propose privacy-preserving SVM based on secure sum of integers (Section 3.1.1) and secure sum of matrices targeted on vertically partitioned data. To achieve complete security, (Yu, Vaidya and Jiang, 2006) suggest to apply the circuit evaluation technique. Subsequently, they extend the existing SVM (Yu, Vaidya and Jiang, 2006) to support arbitrarily partitioned data (Vaidya, Yu and Jiang, 2008). (Laur et al., 2006) use some cryptographic tools (e.g., homomorphic encryption, the secret sharing and the circuit protocol) to implement the kernel adaption and the kernel perception learning algorithms without revealing the Gram (kernel) matrix of the data.

3.2.3 Privacy-Preserving Decision Tree

Decision tree learning is one of many popular methods for inductive inference. The resultant tree can be turned into a set of if-then rules for decision making. The main idea of the decision tree is to split a set of choices about each feature of data in turn, starting first at the root of the tree and then moving down to the leaves. The decision tree uses

a greedy heuristic approach that performs search and evaluates all the possible options at the current stage of learning to find the optimal split point. A few metrics such as information gain and entropy (Quinlan, 1986), gini (Breiman et al., 1984), and gain ratio (Quinlan, 1993) can be used to decide the optimal split point. Decision tree algorithms that include ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993) , CART (Breiman et al., 1984) and many more can be used to construct and evaluate a tree. More details of decision tree induction can be found in (Mitchell, 1997; Quinlan, 1993; Safavian and Landgrebe, 1991).

In the two-party model, Alice and Bob each hold their own data respectively. Alice intends to construct (induce) a decision tree based on both of her own data and the data from Bob without violating data privacy of Bob (and likewise for Bob).

Binary column vectors can represent nominal (categorical) attribute values in decision tree construction. The construction process incurs nodes splitting by evaluating the "goodness" (information gain) of the attributes. The information gain is the entropy of the attribute set minus the entropy when a particular attribute is selected, as follows.

$$\text{Gain}(S, F) = \text{Entropy}(S) - \sum_{\alpha \in F} \left(\frac{|\sigma_{F=\alpha}(S)|}{|S|} \text{Entropy}(\sigma_{F=\alpha}(S)) \right), \tag{3.20}$$

where

$$\text{Entropy}(S) = -\sum_{i=1}^{N} \left(\frac{|\sigma_{C=i}(S)|}{|S|} \log \frac{|\sigma_{C=i}(S)|}{|S|} \right), \tag{3.21}$$

where S is the data set having N classes, α is the value of the attribute F, σ is the selection operator, and C is the class label. For example, in Figure 3.6(a), the class information of the original data is known by Alice and Bob. They individually can compute $\text{Entropy}(S) = -\frac{2}{5}\log(\frac{2}{5}) - \frac{3}{5}\log(\frac{3}{5}) = 0.971$ where the total number of "Yes" and "No" of the class "PlayBall" are 2 and 3, respectively. Let "Outlook" be an attribute that has two values, "Rain" and "Sunny", for information gain calculation. Alice (3.6(b)), and Bob (3.6(c)), jointly compute the entropy of the attribute "Outlook" without revealing their inputs to each other. They apply secure scalar product protocol (Section 3.1.2) to compute the number of tuples in the data set S which the attribute "Outlook" has value "Rain" in the following.

$$\begin{aligned} |\sigma_{[Outlook=\text{“}Rain\text{”}]}(S)| &= \text{Alice}_{[Outlook=\text{“}Rain\text{”}]} \cdot \text{Bob}_{[Outlook=\text{“}Rain\text{”}]} \\ &= [0,1,1,1,1]^T \cdot [1,0,1,1,1]^T = 3. \end{aligned} \tag{3.22}$$

Similarly, $|\sigma_{[Outlook=\text{“}Rain\text{”},PlayBall=\text{“}Yes\text{”}]}|$ and $|\sigma_{[Outlook=\text{“}Rain\text{”},PlayBall=\text{“}No\text{”}]}|$ are 2 and 1 respectively. The entropy of $(\sigma_{[Outlook=\text{“}Rain\text{”}]}(S))$, based on Equation 3.21, is

$$\text{Entropy}(\sigma_{[Outlook=\text{“}Rain\text{”}]}(S)) = -\frac{1}{3}\log(\frac{1}{3}) - \frac{2}{3}\log(\frac{2}{3}) = 0.917.$$

Similarly, the entropy of $(\sigma_{[Outlook=\text{“}Sunny\text{”}]}(S))$ is 0. Thus, the information gain of the attribute "Outlook", based on Equation 3.20, is

$$\text{Gain}(S, Outlook) = 0.971 - \frac{2}{5} \times 0 - \frac{3}{5} \times 0.917 = 0.42. \tag{3.23}$$

The information gain of the attribute "Humidity" (i.e., $\text{Gain}(S, Humidity)$) and of the attribute "Wind" (i.e., $\text{Gain}(S, Wind)$) can be computed in such a similar way of the information gain of the attribute "Outlook". The best splitting attribute is "Outlook" that has higher information gain than the other two attributes. The attribute "Outlook" next splits the data set S into partitions, $\sigma_{Outlook=\text{“}Sunny\text{”}}$ and $\sigma_{Outlook=\text{“}Rain\text{”}}$.

Above the partition $(\sigma_{Outlook=\text{“}Rain\text{”}})$ can be further split, the information gain of $(\sigma_{Outlook=\text{“}Rain\text{”}}(S), Wind)$ and of $(\sigma_{Outlook=\text{“}Rain\text{”}}(S), Humidity)$ can be computed in such a similar way as above to select a higher information gain between them. The process continues until the complete tree is built. More details of the tree construction process can be found in (Vaidya, Clifton, Kantarcioglu and Patterson, 2008; Du and Zhan, 2002).

(Lindell and Pinkas, 2000) propose a privacy-preserving ID3 algorithm that can securely construct a tree from horizontally partitioned data that involves two parties. They propose a secure logarithm (Section 3.1.7) to compute information gain along with some existing cryptographic tools such as the secure scalar product protocol (Section 3.1.2), the oblivious transfer protocol and the garbled circuit approach. (Du and Zhan, 2002) use the secure scalar product protocol (Section 3.1.2) to securely construct a tree using a semi-trusted commodity server. The commodity server helps to compute the scalar product of vertically partitioned data of two parties without learning their inputs. The approach of (Du and Zhan, 2002) is considered secure by assumption that no collusion occurs between the commodity server and either of the two parties. (Vaidya, Clifton, Kantarcioglu and Patterson, 2008) propose a privacy-preserving ID3 that can securely construct a tree from

vertically partitioned data of multiple parties (i.e., ≥ 2) using the secure set intersection cardinality protocol (Section 3.1.4).

3.2.4 Privacy-Preserving Association Rule Mining

We use the formal definition of (Agrawal et al., 1993) to state the association rule mining problem in the following. Each associate rule is implicitly expressed in the form of $X \longrightarrow Y$, where X and Y are disjoint itemsets (i.e., $X \cap Y$). To mine association rules, two common metrics, the *support* and the *confidence* can determine whether the rule $X \longrightarrow Y$ is frequent. The support and the confidence are defined as follows:

$$\text{support}(X \longrightarrow Y) = \frac{|X \cup Y|}{N}, \tag{3.24}$$

$$\text{confidence}(X \longrightarrow Y) = \frac{|X \cup Y|}{|X|}, \tag{3.25}$$

where $|X|$ (resp. $|Y|$) is the total number of items X (resp. Y) appearing in the transactions, and N is the number of transactions.

Any generated association rule needs to satisfy both a user-defined minimum support ($minsupport$) and a user-defined minimum confidence ($minconf$). Association rule generation comes with two separate steps as follows: (i) finding all the frequent itemsets that have the support above $minsupport$, (ii) generating a rule $X \longrightarrow Y$ from the pair $< X >$ and $< X, Y >$ of the frequent itemsets in the previous step that the confidence of the rule is above $minconf$. Many of the association rule mining algorithms have been proposed such as the Apriori pruning algorithm (Agrawal and Srikant, 1994), the AIS algorithm (Agrawal et al., 1993) the FP-tree algorithm (Han et al., 2000), and the SOTrielT algorithm (Das et al., 2001). The details of the theoretical discussions and the empirical comparisons of above algorithms can be found in (Hipp et al., 2000)

A transactional database can be presented in a Boolean $m \times n$ matrix, where m and n are the number of transactions and the number of items, respectively. Value 1 indicates an item that is in the database and 0 otherwise. Let $\overrightarrow{X}$ and $\overrightarrow{Y}$ be columns of the matrix that present items X and Y, held by Alice and Bob respectively. The frequency of items $X \cup Y$ that appears in the transactions can be computed by scalar dot product,

$$|X \cup Y| = \overrightarrow{X} \cdot \overrightarrow{Y}. \tag{3.26}$$

(Kantarcioglu and Kardes, 2006) propose a method that can mine association rules from horizontally partitioned data that involves more than two parties. They integrate SMC to minimize the information leak. In the vertically partitioned data with the two-party model, (Vaidya and Clifton, 2002) use the secure scalar product protocol (Section 3.1.2) to mine association rules. In the protocol, they apply some linear algebra techniques to mask the private vectors with some perturbed numbers. In terms of time complexity, the method of (Vaidya and Clifton, 2002) outperforms other privacy-preserving association rule mining algorithms based on SMC. To support the vertically partitioned data with more than two parties, (Vaidya and Clifton, 2005) apply the secure set intersection cardinality protocol (Section 3.1.4) to mine association rules.

3.2.5 Privacy-Preserving Clustering

Clustering (Han et al., 2006) is the process of grouping data instances into a small number of clusters. The cluster is a collection of data instances that are similar to one another and dissimilar to other data instances in other clusters. In machine learning (Han et al., 2006), clustering is an unsupervised learning that does not depend on training examples. The similarity of data instances can be computed using the distance-based cluster analysis. Other measurements, such as k-means (MacQueen, 1967) and k-medoids, can also measure the similarity of data instances. Several clustering approaches that include the partitioning approach, the hierarchical approach, the density-based approach and the grid-based approach, can be used to cluster data instances.

In the partitioning approach, the partitioning algorithms can construct the predefined partitions of the data. All data instances are likely similar that they are grouped in the same cluster. Well-known partitioning algorithms include k-mean (MacQueen, 1967), k-medoids, expectation maximization (Mitchell, 1997), and some of their variations. The hierarchical approach decomposes data instances by grouping them into a tree of clusters. The hierarchical tree is constructed based on the agglomerative (i.e., bottom-up) or divisive (i.e., top-down) hierarchical clustering. Algorithms of the hierarchical approach include BRICH (Zhang et al., 1996), CURE (Guha et al., 1998) and Chameleon (Karypis et al., 1999). Many partitioning clustering algorithms can use the density-based approach based on the distance between data instances. Algorithms of the density-based approach include DBSCAN (Ester et al., 1996), DENCLUE (Hinneburg and Keim, 1998) and OPTICS

(Agrawal et al., 2003). The grid-based approach quantizes data space into a number of cells that can construct a grid structure. Algorithms of the grid-based approach include STING (Wang et al., 1997), WaveCluster (Sheikholeslami et al., 1998), CLIQUE (Agrawal et al., 1998) and MAFIA (Goil et al., 1999). (Berkhin, 2002) provides a survey paper about clustering. Data mining clustering techniques are discussed in detail in (Tan et al., 2006).

(Vaidya and Clifton, 2003) propose a privacy-preserving k-means clustering to mine association rules from vertically partitioned data that involves more than two parties. Given a sample held by different parties, the method of (Vaidya and Clifton, 2003) can allow multiple parties to jointly and securely compute the sample of which cluster is closest to it. The method uses the secure permutation protocol (Du and Atallah, 2001b) (Section 3.1.5) and the secure comparison protocol that involves the circuit approach (Yao, 1986).

(Jagannathan and Wright, 2005) first introduce the concept of arbitrarily partitioned data in clustering. Both horizontally and vertically partitioned data are considered special cases of the arbitrarily partitioned data. They propose a privacy-preserving k-means clustering targeted on arbitrarily partitioned data. To securely compute the closest cluster for a given instance, privacy-preserving k-means clustering (Jagannathan and Wright, 2005) uses the scalar product protocol (Section 3.1.2). (Jagannathan et al., 2006) also propose a privacy-preserving k-clustering using a simple I/O-efficient algorithm. The algorithm is claimed to be more accurate than others using the iterative k-means algorithm to produce cluster centers. Thus, (Jagannathan et al., 2006) apply the secure scalar product protocol (Section 3.1.2) and the circuit approach (Yao, 1986) to mine association rules.

(Lin et al., 2005) apply EM (expectation maximization) mixture modeling to securely compute clustering over horizontally partitioned data. Each party computes partitions locally on its input data. All parties then apply the secure sum protocol (Section 3.1.1) to compute the global sum of their partitions without revealing any individual input to each other. (Prasad and Rangan, 2006) propose a privacy-preserving BRICH algorithm to perform clustering targeted on vertically partitioned data. They use various secure building blocks such as the secure scalar product protocol (Section 3.1.2) and the random permutation protocol, to mine association rules.

3.2.6 Other Privacy-Preserving Data Mining Algorithms

(Du et al., 2004) propose the privacy-preserving approach of multivariate classification and that of multivariate linear regression over vertically partitioned data that involves two parties. They apply a practical security model that consists of a number of secure building blocks, such as the secure matrix inverse protocol and the secure matrix product protocol. Privacy-preserving neural network (Barni et al., 2006) allows two parties under the provided scheme that one of two parties holds its own private data and another has a private neural network that can process the private data. (Naor and Pinkas, 1999) apply some cryptographic tools such as the secure scalar product protocol and the private polynomial evaluation (Section 3.1.6), to the privacy-preserving neural network.

(Brickell et al., 2007) propose a privacy-preserving evaluation protocol for diagnostic programs that can present in binary trees or branching programs. The protocol uses classification labels in the branching diagnostic program. At the end of the protocol execution, the participating parties learn the label without knowing the diagnostic program. The owner of the diagnostic program learns nothing. (Sakuma et al., 2008) enhance a reinforcement learning technique with privacy-preserving enabled. The learning protocol can discover a control policy via interactions between distributed agents without revealing any agent information to each other. The protocol uses secure building blocks that include the secure comparison protocol and the secure division protocol based on the homomorphic encryption scheme. (Agrawal et al., 2003) propose sharing information across autonomous entities that can provide an answer to a query. They propose a few secure building blocks that include the commutative encryption and the hash functions. To reveal the answer of the query only, (Agrawal et al., 2003) use the secure building blocks to propose a few secure protocols in computing intersection and equijoin.

3.2.7 Discussion

In privacy-preserving data mining (PPDM), the randomization and SMC can perform secure computation in many data mining tasks. (Teng and Du, 2007) propose a hybrid approach to combine above two approaches to build a decision tree. They first apply the randomization to select higher information gains of some attributes. Next, the best information gain is discovered using the cryptographic approach (SMC). (Teng and Du,

2007) uses the randomization that can help to reduce expensive cost in the cryptographic approach. However, the method does not provided a complete privacy because the method allows some information leaks.

Many existing PPDM algorithms assume that all participating parties are semi-honest: they strictly follow the protocol and will not collude with each other, but are curious in inferring additional knowledge using polynomial-time computations. However, (Kantarcioglu and Kardes, 2006) use the zero-knowledge proof to detect malicious behaviors in the secure scalar product protocol. (Jiang and Clifton, 2007) also propose the Accountable Computing (AC) framework that relies on an independent entity (i.e., without contributing any data) to detect malicious behaviors in the secure scalar product protocol. To withstand probing attacks from a malicious party, (Vaidya and Clifton, 2005) propose a method in the privacy-preserving association rule mining. All above discussed methods can withstand specific attacks from the malicious party in the secure scalar product protocol. The PPDM algorithms based on the malicious model incur expensive cost in the modulus exponentiations and the communication cost. In many situations, PPDM based on the semi-honest model is sufficient for many data mining tasks (Aggarwal and Yu, 2008). Therefore, we will investigate PPDM based on SMC under the semi-honest model in our thesis. We aim to propose SMC protocols that can provide complete privacy and also be efficient in computation and effective in protecting data privacy in PPDM.

In secure multi-party computation, many PPDM algorithms apply secure building blocks (Section 3.1) to perform secure computations, such as Bayesian network structure (Wright and Yang, 2004), Naïve Bayes (Vaidya and Clifton, 2004) association rule mining (Vaidya and Clifton, 2002), decision tree (Lindell and Pinkas, 2000), k-means (Jagannathan and Wright, 2005; Vaidya and Clifton, 2003), support vector machine (Laur et al., 2006), singular value decomposition (Han et al., 2009) and many more. However, many privacy-preserving data mining algorithms based on SMC are ad-hoc - they are proposed for specific data mining tasks, and thus cannot be applied to other tasks directly. To address the current limitation, we will investigate a general SMC model for privacy-preserving data mining algorithms.

Secure log (Section 3.1.7) and secure division (Section 3.1.8) are important operators in many data mining tasks. As discussed in Sections 3.1.7 and 3.1.8, many of them are not efficient in computation or not effective in protecting data privacy, and both of them. We

will propose a secure log operator and a secure division operator. They are SMC protocols that can be efficient in computation and effective to protect data privacy. In PPDM, many secure building blocks (Section 3.1) based on the circuit approach (Sections 2.5.1 and 2.5.2) can not compute many functions in data mining primitives. The circuit approach also has prohibitive computation in many data mining tasks. In this thesis, we focus to use a hybrid model that combines the homomorphic encryption protocol and the circuit approach.

In this thesis, we will propose a general model for privacy-preserving data mining. In our model, we propose a set of secure operators to compute many functions in data mining primitives. All secure operators can also be pipelined together to perform various tasks in data mining. To achieve complete privacy and be efficient in computation, we use the hybrid model that combines the circuit approach and the homomorphic encryption protocol in PPDM. As case studies in data mining, we will apply our general model into some data mining tasks. Our general model for privacy-preserving data mining can even support wider applications. Thus, as a case study in other application domain, we will also apply the proposed model into the optimization problem and the traveling salesman problem (TSP).

3.3 Other Privacy Preservation Techniques

We have discussed the privacy preservation techniques in privacy-preserving data mining (PPDM). In this section, we discuss some other privacy preservation techniques.

- **The k-anonymity model and ℓ-diversity.** An adversary can use indirect identifications from public databases to identify individual records; e.g., combining record attributes of public databases can possibly identify individual records. The k-anonymity model (Samarati, 2001) can address the issue by applying some techniques, such as generalization and suppression, so as to reduce the granularity of data representation. To reduce the data granularity, any record must be mapped onto at least k other records. The generalization can make the attribute values into a range value, e.g., converting date of birth into year of birth. In the suppression method, it removes the attribute values completely. Therefore, the k-anonymity method can protect identities to the level of k-individuals. The method can leak the corresponding sensitive values when the homogeneity of the values exists within a

group. (Machanavajjhala et al., 2007) propose ℓ-diversity to address some limitations in the k-anonymity method. In the ℓ-diversity, the intra-group diversity of sensitive values is bounded within the anonymization scheme. Thus, the ℓ-diversity can maintain a minimum group size of k in the k-anonymity method with the diversity of the sensitive attributes.

- **Downgrading application effectiveness.** In some situations, the outputs of data mining applications such as classification, associate rule mining and query processing can result in privacy violation regardless of data availability. These lead to research in downgrading application effectiveness. Two common methods, data modification and application modification, are proposed to downgrade the effectiveness in applications, such as classifier downgrading (Moskowitz and Chang, 2000), rule hiding (Verykios, Elmagarmid, Bertino, Saygin and Dasseni, 2004) and query auditing (Adam and Wortmann, 1989).

Chapter 4

DAG: A General Model for Privacy-Preserving Data Mining

In many real-world applications, data are distributed across multiple parties. These parties have a strong willingness of sharing their data, so as to have a global view of the data that cannot be mined from the data of any single party. However, data may contain sensitive information; directly sharing it could violate personal privacy (Aggarwal and Yu, 2008; Clifton et al., 2002). Consider two hospitals, which are interested in building a diagnosis model over their integrated patient records to predict the probability of a person getting diabetes. Such a model usually has better predictive performance than a model built on the dataset of single hospital, since the integrated data contains more information. However, patient records are highly sensitive – they could put the privacy of patients at risk when inappropriately disclosed. Government agencies have enacted laws to protect personal privacy. Well known representatives are HIPAA (Congress, 1996) of the Unites States and ECHR (ECHR, 2014) of the European Union. Therefore, for the sake of personal privacy and also to follow the laws, data sharing for analytics needs to be carried out in a privacy-preserving way.

Secure multi-party computation (SMC) is a fundamental field in cryptography. It allows multiple parties to jointly compute a function over their inputs, while keeping respective inputs of all the parties private. Such a property is very effective in protecting personal privacy. Thus, SMC has been extensively applied in privacy-preserving data mining (PPDM), such as in decision tree (Vaidya, Clifton, Kantarcioglu and Patterson, 2008), Naïve Bayes (Vaidya, Kantarcioglu and Clifton, 2008) and support vector machine (Teo et al., 2013; Yu, Jiang and Vaidya, 2006). However, these proposed solutions are ad-hoc and specific to tasks. They cannot be directly applied to other tasks. Many SMC solutions also provide a limited set of secure operators.

To address the above issues, we propose an SMC-based model DAG (Directed Acyclic Graph). It can be generally applied in privacy-preserving functions involving multiple parties. Our model is based on an observation. That is, many functions, even those complicated ones in data mining like support vector machine, ID3 and Naïve Bayes, can be decomposed into a set of basic operations, such as $+$, $-$, $\times$, and $/$. When these basic operations are pipelined accordingly, a workflow system is formed, and various functions can be implemented. The pipeline of operators forms a directed acyclic graph. Therefore, we name our model as DAG.

Our DAG model consists of 3 types of nodes: *source*, *sink*, and *operator*. Source nodes are private inputs of the parties involved in the tasks. Sink nodes are the outputs of the model. Operators are basic operations, e.g., $\times$, $/$, and log (logarithm). The nodes in the model are connected by directed edges, which represent data flow, i.e., the outputs of the upper stream nodes are the inputs of the downstream nodes. To keep the respective input of each party confidential, security requirements are enforced on the operators. Specifically, we formulate each operator as an SMC protocol (Yao, 1986), such that given the private inputs of multiple parties, the operator allows the parties to learn the operator output while keeping the inputs confidential.

Thus far, our DAG model consists of 9 operators such as secure addition (Section 4.1.1), secure minus (Section 4.1.2), secure multiplication (Section 4.1.3), secure division (Section 4.1.5), secure log (Section 4.1.6), secure power (Section 4.1.7), secure bit-length (Section 4.1.4), secure max (Section 4.1.8), and secure max location (Section 4.1.9). The functions it can support are these 9 operators, and their compositions. Still, our model is extendable. Other basic operators, such as secure sine and secure cosine, can be defined and added to our model, such that more functions can be supported. Section 4.1 provides the details of the model.

We theoretically analyze our DAG model. The security of every single operator can be proven by simulation paradigm (Goldreich, 2004), a standard methodology of proving the security of SMC protocols. We also prove the security of the model when operators are pipelined to serve the desired functions. Cryptography supports big integers, while inputs in real applications can be floating values. As such, we set necessary parameters and apply various techniques, like Taylor series, to preserve the precision of operator output. We provide the error bound of every single operator, and also the error bound of the connection

of operators. It turns out when two operators are connected, their accumulated error is bounded by the summation of their respective errors. Furthermore, every operator as an SMC protocol has its complexity. We thus also analyze for every operator the time complexity and communication complexity.

In the following, we first propose our DAG model in Section 4.1. The model analysis is discussed in Section 4.2. We evaluate the performance of the model in Section 4.3. Lastly, we summarize this chapter in Section 4.4

4.1 Directed Acyclic Model (DAG)

In this section we present our DAG model. We consider two parties: Alice and Bob. We assume a *semi-honest* setting. That is, Alice and Bob are *honest-but-curious* – they strictly follow the protocol and will not collude with each other but are interested in inferring additional knowledge using polynomial-time computations.

The DAG model consists of 3 types of nodes: source, sink, and secure operator. The source nodes are the sensitive data of Alice and Bob, and are the inputs of the DAG model. The sink nodes are the private outputs of the model, and are distributed to Alice and Bob. A secure operator is a private function, which allows Alice and Bob to compute a function of their respective inputs, while keeping all the inputs confidential.

Definition 4.1 (Secure Operator) *Let $A = (a_1, a_2, \ldots, a_l)$ and $B = (b_1, b_2, \ldots, b_m)$ be the private inputs of Alice and Bob, respectively, where a_i and b_j are integers for $i = 1, 2, \ldots l$ and $j = 1, 2, \ldots, m$. Secure operator keeps A and B confidential, and computes a function $f : (A, B) \rightarrow c_1 + c_2$, where c_1 is known only by Alice and c_2 is known only by Bob.*

In the above definition, c_1 and c_2 are distributed to Alice and Bob, respectively. If the two parties need to know the function output, they release c_1 and c_2.

Secure operators are connected by edges in the DAG model. The outputs of the upstream secure operators are the inputs of the downstream ones. We assume that there is no cycle in our model. In this section, we propose 9 basic operators: secure addition (Section 4.1.1), secure minus (Section 4.1.2), secure multiplication (Section 4.1.3), secure division (Section 4.1.5), secure log (Section 4.1.6), secure power (Section 4.1.7), secure bit-length (Section 4.1.4), secure max (Section 4.1.8), and secure max location (Section

4.1.9) according to Definition 4.1. The secure operators are essentially secure multi-party computation (SMC) protocols between Alice and Bob. Our DAG model is extendable – other secure operators satisfying Definition 4.1 can be easily integrated.

4.1.1 Secure Addition

Let $A = (a_1, a_2, \ldots, a_l)$ and $B = (b_1, b_2, \ldots, b_m)$ be the private inputs of Alice and Bob, respectively. The secure addition of A and B equal to $\sum_{i=1}^{l} a_i + \sum_{j=1}^{m} b_j$ is straightforward. Alice independently computes $c_1 = \sum_{i=1}^{l} a_i$ and Bob independently computes $c_2 = \sum_{j=1}^{m} b_j$. At the end of the execution, Alice and Bob hold c_1 and c_2 respectively, where $c_1 + c_2 = \sum_{i=1}^{l} a_i + \sum_{j=m}^{n} b_j$.

4.1.1.1 The Protocol Analysis

We now analyze secure addition for its complexity and security.

Time Complexity. We measure the time complexity of secure addition by modular exponentiations, since they consume most of the time. Alice and Bob can compute their respective inputs locally to generate c_1 and c_2, held by Alice and Bob respectively. Therefore they use no modular exponentiation.

Communication Complexity. We measure the communication complexity by the number of message bits passing between Alice and Bob. Again, Alice and Bob can compute their respective inputs locally to generate c_1 and c_2, held by Alice and Bob respectively. Therefore no interaction (i.e., no communication cost) is required between Alice and Bob.

Security. Since Alice and Bob transfer nothing to each other, secure addition protocol is secure.

4.1.2 Secure Minus

Let $A = (a_1, a_2, \ldots, a_l)$ and $B = (b_1, b_2, \ldots, b_m)$ be the private inputs of Alice and Bob, respectively. The secure minus of B from A equal to $\sum_{i=1}^{l} a_i - \sum_{j=1}^{m} b_j$ is as follows. Specifically, Alice computes $c_1 = \sum_{i=1}^{l} a_i$, and Bob computes $c_2 = -\sum_{j=1}^{m} b_j$. At the end of the execution, Alice and Bob hold c_1 and c_2 respectively, where $c_1 + c_2 = \sum_{i=1}^{l} a_i - \sum_{j=1}^{m} b_j$.

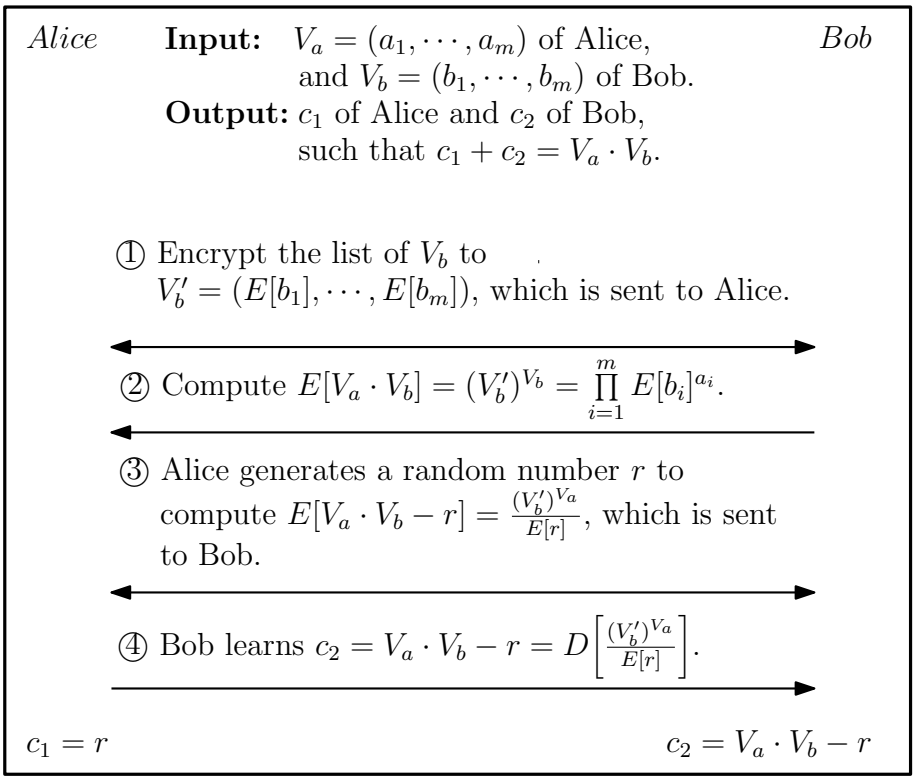


Figure 4.1: Secure multiplication protocol

Time Complexity. We measure the time complexity of secure minus by modular exponentiations, since they consume most of the time. Alice and Bob can compute their respective inputs locally to generate c_1 and c_2, held by Alice and Bob respectively. Therefore they use no modular exponentiation.

4.1.2.1 The Protocol Analysis

We now analyze secure minus for its complexity and security.

Communication Complexity. We measure the communication complexity by the number of message bits passing between Alice and Bob. Again, Alice and Bob can compute their respective inputs locally to generate c_1 and c_2, held by Alice and Bob respectively. Therefore no interaction (i.e., no communication cost) is required between Alice and Bob.

Security. Since Alice and Bob transfer nothing to each other, secure minus protocol is secure.

4.1.3 Secure Multiplication

Let $V_a = (a_1, a_2, \ldots, a_m)$ and $V_b = (b_1, b_2, \ldots, b_m)$ be the private inputs of Alice and Bob, respectively. Secure Multiplication keeps V_b confidential to Alice and V_a confidential to Bob, and applies the Secure Scalar Product (SSP) protocol (Goethals et al., 2004) to compute c_1 and c_2, such that $c_1 + c_2 = \sum_{i=1}^{m} a_i \times b_i$. At the end of the protocol, Alice learns c_1 but not c_2, and Bob learns c_2 but not c_1.

The Protocol of Secure Multiplication. We present the secure multi-party (SMC) protocol of secure multiplication in Figure 4.1. Bob configures Paillier cryptosystem to generate a pair of keys (pk, sk), which sk is the secret key and pk is the public key. Let $E[.]$ and $D[.]$ be the encryption and decryption functions corresponding to pk and sk, respectively. The public key pk is sent to Alice. Alice and Bob carry out the protocol step by step as follows.

Step 1. Let $V_a = (a_1, a_2, \ldots, a_m)$ and $V_b = (b_1, b_2, \ldots, b_m)$ be the private inputs of Alice and Bob, respectively. Bob encrypts the list of V_b to $V_b' = (E[b_1], \cdots, E[b_m])$, which is sent to Alice.

Step 2. Alice computes the encrypted list with her private input V_a as follows,

$$(V_b')^{V_a} = \prod_{i=1}^{m} E[b_i]^{a_i}. \tag{4.1}$$

Step 3. Alice generates a random number $r \in [L_\ell, L_u + B - 1]$, where L_ℓ and L_u are the lower and upper bounds of $V_a \cdot V_b$, $B = 2^{\lambda + \lceil \log(L_u - L_\ell) \rceil + 1}$, and λ is a threshold. She then computes

$$E[V_a \cdot V_b - r] = \frac{(V_b')^{V_a}}{E[r]}, \tag{4.2}$$

which is sent to Bob.

Step 4. Bob uses his secret key sk to decrypt

$$c_2 = D\left[\frac{(V_b')^{V_a}}{E[r]}\right] = V_a \cdot V_b - r, \tag{4.3}$$

where $c_2 \in [L_\ell - L_u - B + 1, L_u - L_\ell]$. Alice sets $c_1 = r$ and Bob sets $c_2 = V_a \cdot V_b - r$ where $c_1 + c_2 = V_a \cdot V_b$.

Note. The value of c_2 can be negative. However, after the modulo n (i.e., mod n) operation by the Paillier decryption (where n is a parameter equal to the product of two big prime numbers), it is always positive. To recover the value of c_2 correctly, we require that every positive integer in our model is less than $n/2$. Thus, if an output (let say c_2) of Paillier decryption is bigger than $n/2$, it must be negative with the value actually equal to $c_2 - n$.

As defined above, the c_1 value is a random number. Thus, it is statistically indistinguishable from any random number in $[L_\ell, L_u+B-1]$. Next, we prove that the distribution of c_2 is also statistically indistinguishable from that of a random number.

Lemma 4.1 *Let r be a random value in $[L_\ell - L_u - B + 1, L_u - L_\ell]$. Then, the statistical distance between c_2 and r is smaller than $\frac{1}{2^\lambda}$.*

Proof 4.1 *Denote $k = V_a \cdot V_b$. Then, $c_2 = k - c_1$. The statistical distance between c_2 and r by Definition 2.2 is*

$$\Delta(r, c_2) = \frac{1}{2} \sum_{v=L_\ell - L_u - B + 1}^{L_u - L_\ell} \left| \Pr[r = v] - \Pr[k - c_1 = v] \right|. \tag{4.4}$$

We know that k must fall in $\Omega = [L_\ell, L_u]$. For any v we have

$$\begin{aligned} \Pr[k - c_1 = v] &= \sum_{m \in \Omega} \Pr[k = m] \Pr[c_1 = m - v] \\ &\leq \frac{1}{(L_u - L_\ell + B)} \sum_{m \in \Omega} \Pr[k = m] = \frac{1}{L_u - L_\ell + B}. \end{aligned}$$

If $-B+1 \leq v \leq 0$, then $\Pr[x - c_1 = v] = \frac{1}{L_u - L_\ell + B}$. We also know $\Pr[r = v] = \frac{1}{2(L_u - L_\ell) + B}$ for $L_\ell - L_u - B + 1 \leq v \leq L_u - L_\ell$, and $\Pr[r = v] = 0$ otherwise. Therefore,

$$\begin{aligned} \Delta(r, c_2) &\leq \frac{1}{2} \left(\sum_{v=L_\ell - L_u - B + 1}^{-B} \left| \frac{1}{2(L_u - L_\ell) + B} - 0 \right| \right. \\ &\quad \left. + \sum_{v=-B+1}^{0} \left| \frac{1}{2(L_u - L_\ell) + B} - \frac{1}{L_u + B - L_\ell} \right| + \sum_{v=1}^{L_u - L_\ell} \left| \frac{1}{2(L_u - L_\ell) + B} - 0 \right| \right) \\ &< \frac{2(L_u - L_\ell)}{B} = \frac{1}{2^\lambda}, \end{aligned}$$

where the last equation holds since $B = 2^{\lambda + \lceil \log(L_u - L_\ell) \rceil + 1}$.

We set λ to be a value, such that $\frac{1}{2^\lambda}$ is negligible. The above lemma proves that c_2 and r are actually statistically indistinguishable. The randomness of c_1 and c_2 ensures that each value does not disclose any information. Such a property is important, when the output of secure multiplication is input to other operators. Then, the output of secure multiplication is intermediate, and should disclose no information.

In Definition 4.1 we specify that the output of every secure operator is split into two portions c_1 and c_2. Their range settings and value computations are similar to those of secure multiplication above. Therefore, in the discussion of the following secure operators we will omit the details as well as the security proof.

4.1.3.1 The Protocol Analysis

We now analyze secure multiplication on the approximation error, complexity, and security.

Floating Value to Integer Conversion. In secure multiplication protocol, the values need to be of integers. This is the standard configuration in an SMC protocol. However, the values of Alice $a_i \in A$ and Bob $b_i \in B$ can be floating values. They thus need to be rounded to integers. To preserve the precision, a_i and b_i are multiplied by a big integer τ and then rounded, i.e., $a = \lfloor a_i \times \tau \rfloor$ and $b = \lfloor b_i \times \tau \rfloor$. Then, a and b are taken as the inputs of the secure multiplication. Let c_1 and c_2 be the respective outputs of Alice and Bob from the secure multiplication. Both c_1 and c_2 are divided by τ to get the final results. In the following of the thesis if any secure operator's input is floating value, we adopt similar operation as the above. The following lemma gives the error bound of secure multiplication due to value rounding.

Lemma 4.2 *Let a_i and b_i be two values larger than or equal to $2^{-\gamma}$, and $\tau = 2^{\gamma+\beta+1}$, where γ and β are positive integers. Let $a = \lfloor a_i \times \tau \rfloor$ and $b = \lfloor b_i \times \tau \rfloor$. Then, the relative error by value rounding in secure multiplication is at most $2^{-\beta}$.*

Proof 4.2 *Based on the rounding, it follows that $a \geq a_i \times \tau - 1$ and $b \geq b_i \times \tau - 1$. Thus, the relative error is*

$$\frac{a_i \times b_i - \frac{a \times b}{\tau^2}}{a_i \times b_i} = \frac{(a_i \times \tau) \cdot (b_i \times \tau) - a \times b}{(a_i \times \tau) \cdot (b_i \times \tau)} < \frac{(a_i \times \tau) + (b_i \times \tau)}{(a_i \times \tau) \cdot (b_i \times \tau)} = \frac{1}{a_i \times \tau} + \frac{1}{b_i \times \tau}.$$

According to the preconditions in the lemma, both $a_i \times \tau$ *and* $b_i \times \tau$ *are at least* $2^{\beta+1}$. *Therefore,*

$$\frac{1}{a_i \times \tau} + \frac{1}{b_i \times \tau} \leq 2^{-\beta},$$

which concludes the proof.

Time Complexity. We measure the time complexity of secure multiplication by modular exponentiations, since they consume most of the time. Secure multiplication in Step 1 and 2 requires $3m$ modular exponentiations. The other 2 steps together take 3 modular exponentiations. Therefore, the number of modular exponentiations needed by secure multiplication is $3(m+1)$ bounded by $O(m)$.

Communication Complexity. We measure the communication complexity by the number of message bits passing between Alice and Bob. The communication cost of Step 1 and 2 is $2t_2m$ bits where t_2 is the message length in Paillier cryptosystem (e.g., $t_2 = 1024$). The other 2 steps is $2t_2$ bits. Therefore, the communication complexity is $2t_2(m+1)$ bounded by $O(m)$.

Secure multiplication is proven secure via simulation paradigm (Section 2.6.1) in the following.

Theorem 4.1 *The secure multiplication protocol is simulatable.*

Proof 4.3 *We simulate the view of Alice and that of Bob. We first simulate the view of Alice. Let* V_a *be the input of Alice, and* c_1 *be the protocol output to her. According to the secure multiplication protocol in Figure 4.1, the view of Alice is* $VIEW_1^{\text{mul}} = (V_a, \mathcal{V}_1)$, *where* $\mathcal{V}_1$ *is the set of encrypted messages she receives to compute* $(V_b')^{V_a}$ *(in Step 2). The simulator* $S_1^{\text{mul}}(V_a, c_1)$ *to simulate* $VIEW_1^{\text{mul}}$ *is created as follows. Each message* $E[m] \in \mathcal{V}_1$ *is a ciphertext of Paillier encryption. To simulate it,* S_1^{mul} *selects a random value* r *and computes* $E[r]$. *Because Paillier encryption is semantically secure,* $E[m]$ *and* $E[r]$ *are computationally indistinguishable.*

The view of Bob is simulated in the following. Let V_b *be the input of Bob, and* c_2 *be the protocol output to him. According to the secure multiplication protocol in Figure 4.1, the view of Bob is* $VIEW_2^{\text{mul}} = (V_b, \mathcal{V}_2)$, *where* $\mathcal{V}_2$ *is the set of messages he receives from Alice for Paillier decryption in Step 3. The simulator* $S_2^{\text{mul}}(V_b, c_2)$ *to simulate* $VIEW_2^{\text{mul}}$

is created as follows. The simulation of $\mathcal{V}_2$ is already given in (Goethals et al., 2004). Thus, S_2^{mul} can call the simulator in (Goethals et al., 2004) to simulate $\mathcal{V}_2$.

4.1.4 Secure Bit-Length

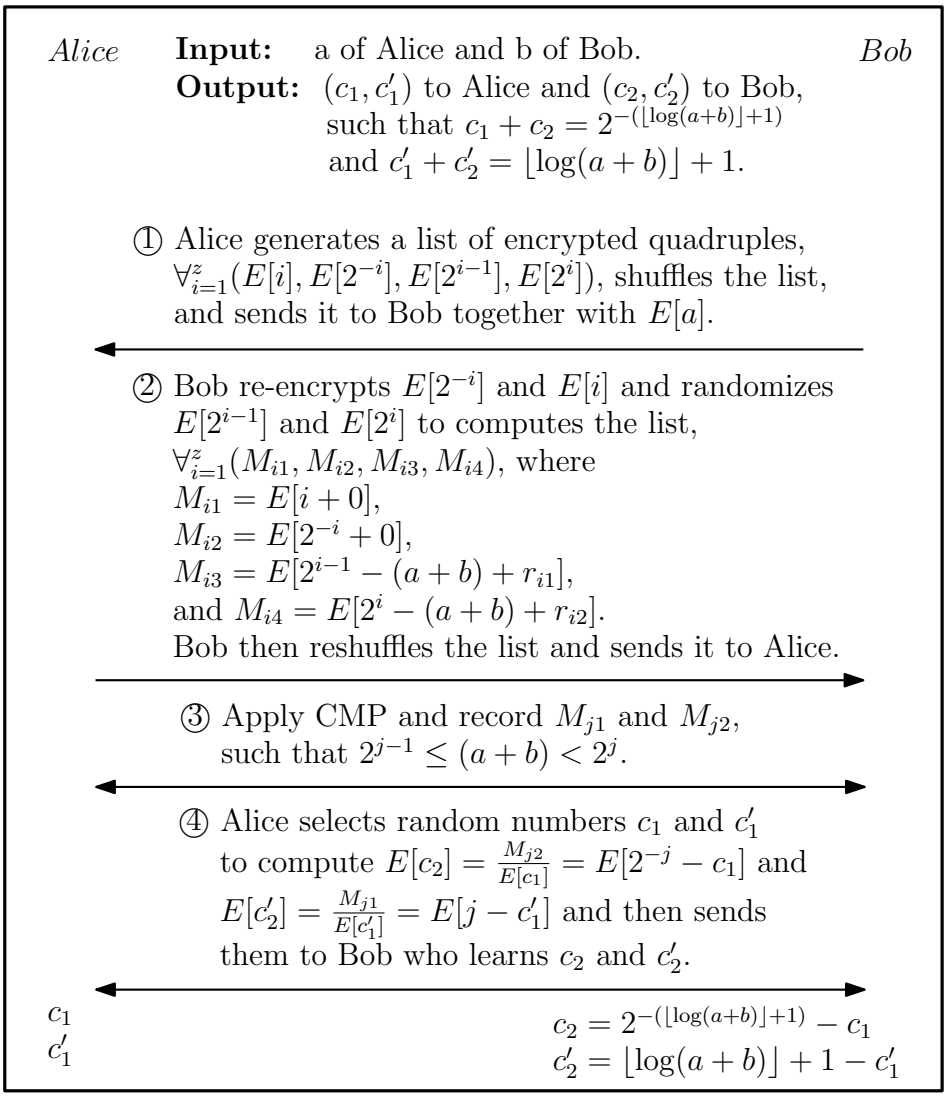


Figure 4.2: Secure bit-length protocol

Figure 4.2 gives the SMC protocol of secure bit-length. Let a and b be the respetive private inputs of Alice and Bob and $a+b > 0$. The protocol outputs c_1, c_1', c_2 and c_2', such that $c_1 + c_2 = 2^{-(\lfloor \log(a+b) \rfloor + 1)}$ and $c_1' + c_2' = \lfloor \log(a+b) \rfloor + 1$. At the end of the protocol, Alice only learns c_1 and c_1' but not b, c_2 or c_2' , and Bob only learns c_2 and c_2' but not a, c_1 or c_1'. Most importantly, the protocol is to find an integer j satisfying $2^{j-1} \leq (a+b) < 2^j$. For such a j, clearly, $j = \lfloor \log(a+b) \rfloor + 1$ and $2^{-j} = 2^{-(\lfloor \log(a+b) \rfloor + 1)}$.

Alice and Bob first configure a (2,2)-threshold Paillier cryptosystem. Let (pk, sk) be the public and private key pair of the cryptosystem. Suppose that sk_A and sk_B are the

secret shares of Alice and Bob respectively, such that the shares combined together can recover sk. Let $E[.]$ and $D[.]$ be the encryption and decryption functions corresponding to pk, and (sk_A, sk_B), respectively. Alice and Bob carry out the protocol step by step as follows.

Step 1. Let $\max(a+b)$ be the maximum value of $a+b$, and $z = \lfloor \log(\max(a+b)) \rfloor + 1$. Alice first generates a list of encrypted quadruples, $\forall_{i=1}^{z}(E[i], E[2^{-i}], E[2^{i-1}], E[2^i])$. She shuffles the position of the quadruples in the list. She then computes $E[a]$, and sends the shuffled list together with $E[a]$ to Bob.

Step 2. For each quadruple $E[i], E[2^{-i}], E[2^{i-1}], E[2^i])$ in the list, Bob re-encrypts $E[2^{-i}]$ and $E[i]$, and randomizes $E[2^{i-1}]$ and $E[2^i]$ by

$$\begin{cases} M_{i1} = E[i+0] = E[i] \times E[0], \\ M_{i2} = E[2^{-i}+0] = E[2^{-i}] \times E[0], \\ M_{i3} = E[2^{i-1} + r_{i1} - (a+b)] = \frac{E[2^{i-1}] \times E[r_{i1}]}{E[a] \times E[b]}, \\ M_{i4} = E[2^{i} + r_{i2} - (a+b)] = \frac{E[2^{i}] \times E[r_{i1}]}{E[a] \times E[b]}, \end{cases} \tag{4.5}$$

where random numbers r_{i1} and r_{i2} are from $\{2^z, 2^z+1, \ldots, 2^\eta - 1\}$, $\eta = \chi + z + 1$, and χ is a threshold. He shuffles the positions of quadruples $\forall_{i=1}^{z}(M_{i1}, M_{i2}, M_{i3}, M_{i4})$, and sends it back to Alice.

The re-encryption and shuffling ensure that neither Alice nor Bob is able to link $(M_{i1}, M_{i2}, M_{i3}, M_{i4})$ with i with a confidence significantly higher than $1/z$. This is guaranteed by the semantic security of Paillier cryptosystem.

Step 3. Alice and Bob jointly decrypt M_{i3} and M_{i4}, and only Alice learns the decrypted results, $2^{i-1} + r_{i1} - (a+b)$ and $2^i + r_{i2} - (a+b)$ for $i = 1, 2, \ldots, z$. Then, Alice and Bob apply CMP (i.e., the secure integer comparison circuit in Section 3.1.10) to compare the decrypted results with (r_{i1}, r_{i2}). Alice records M_{j1} and M_{j2}, where

$$\begin{cases} 2^{j-1} + r_{j1} - (a+b) \leq r_{j1} \\ 2^{j} + r_{j1} - (a+b) > r_{j2}. \end{cases} \tag{4.6}$$

Obviously, $2^{j-1} \leq (a+b) < 2^j$ and $j = \lfloor \log(a+b) \rfloor + 1$.

Step 4. Alice randomly selects c_1 and c_1' and computes $E[c_2] = E[2^{-j} - c_1] = \frac{M_{j2}}{E[c_1]}$ and $E[c_2'] = E[j - c_1'] = \frac{M_{j1}}{E[c_1']}$, which are sent to Bob. The two parties jointly decrypt $E[c_2]$ and $E[c_2']$, and only Bob learns (c_2, c_2'). The statistically indistinguishability of c_1, that of c_1', that of c_2, and that of c_2' can be proven by using a similar proof as in Lemma 4.1.

The correctness of the protocol can be easily verified, since $j = \lfloor \log(a+b) \rfloor + 1$ by Inequalities 4.6. Next, we prove that the randomization in Step 2 is done appropriately. It makes sure that $2^{i-1} - (a+b) + r_{i1}$ and $2^i - (a+b) + r_{i2}$ are indistinguishable from random values.

Lemma 4.3 *Let $e_i = 2^i + r_{i2} - (a+b)$ be the decryption of M_{i3} in secure bit-length protocol, where $z = \lfloor \log(\max(a+b)) \rfloor + 1$, $r_{i2} \in \{2^z, 2^z+1, \ldots, 2^z + 2^\eta - 1\}$, and $i = 1, 2, \ldots, z$. Let r be a random value in $\{2^z, 2^z+1, \ldots, 2^z+2^\eta-1\}$. Then, the statistical distance between e_i and r is smaller than $\frac{1}{2^\lambda}$.*

Proof 4.4 *Denote $k_i = 2^i - (a+b)$. The statistical distance between e_i and r by Definition 2.2 is*

$$\Delta(r, e_i) = \frac{1}{2} \sum_{v=2}^{2^{z+1}+2^\eta-2} \left| \Pr[r = v] - \Pr[k_i + r_{i2} = v] \right|. \tag{4.7}$$

We know that k_i must fall in $\Omega = \{2 - 2^z, 3 - 2^z, \ldots, 2^z - 1\}$. For any v we have

$$\Pr[k_i + r_{i2} = v] = \sum_{m \in \Omega} \Pr[k_i = m] \Pr[r_{i2} = v - m] \leq \frac{1}{2^\eta} \sum_{m \in \Omega} \Pr[k_i = m] = \frac{1}{2^\eta}.$$

If $2^{z+1} - 1 \leq v \leq 2^\eta + 1$, then $\Pr[k_i + r_{i2} = v] = \frac{1}{2^\eta}$. In addition, we know $\Pr[r = v] = \frac{1}{2^\eta}$ for $2^z \leq v \leq 2^z + 2^\eta - 1$, and $\Pr[r = v] = 0$ for any other v value. Therefore,

$$\begin{aligned} \Delta(r, e_i) \leq \frac{1}{2} \Bigg(& \sum_{v=2}^{2^z-1} \left| 0 - \frac{1}{2^\eta} \right| + \sum_{v=2^z}^{2^{z+1}-2} \left| \frac{1}{2^\eta} - 0 \right| + \sum_{v=2^{z+1}-1}^{2^\eta+1} \left| \frac{1}{2^\eta} - \frac{1}{2^\eta} \right| \\ & + \sum_{v=2^\eta+2}^{2^z+2^\eta-1} \left| \frac{1}{2^\eta} - 0 \right| + \sum_{v=2^z+2^\eta}^{2^{z+1}+2^\eta-2} \left| 0 - \frac{1}{2^\eta} \right| \Bigg) < \frac{2^{z+1}}{2^{\lambda+z+1}} = \frac{1}{2^\lambda}, \end{aligned}$$

where the last equation holds since $\eta = \lambda + z + 1$.

We set λ to be a value, such that $\frac{1}{2^\lambda}$ is negligible. Therefore, the above lemma tells that e_i and r are actually statistically indistinguishable. In a similar way we can prove that the statistical distance between $2^{i-1} - (a+b) + r_{i1}$ and $r \in \{2^z, 2^z+1, \ldots, 2^{\lambda+z}\}$

is upper bounded by $\frac{1}{2^\lambda}$. Since the proof is essentially the same as the above lemma, we omit it here.

4.1.4.1 The Protocol Analysis

Time Complexity. We measure the time complexity by modular exponentiations, since they consume most of the time. Steps 1 and 2 need a total of $16z + 2$ modular exponentiations. Step 3 requires $20z$ modular exponentiations. The initialization of CMP (Ishai et al., 2003; Naor and Pinkas, 2001) also takes some modular exponentiations. However, the initialization can be done before the protocol, and its cost can be amortized over all the runnings of secure bit-length. Thus, we do not count its cost in Step 3. Step 4 needs 24 modular exponentiations. The number of modular exponentiations in secure bit-length is thus $36z + 26$ bounded by $O(z)$.

Steps 1, 2, 3 can be run in parallel – we can process each quadruple by a thread. In such a way, the running time of the protocol can be reduced by a factor of z. We have implemented the parallel processing in the experiments.

Communication Complexity. We measure the communication complexity by the number of message bits passing between Alice and Bob. Steps 1 and 2 transfer $16t_2z+2t_2$ bits, where t_2 is the message length in Pailliar cryptosystem. Step 3 needs $6(\lambda+z+2)t_1z+8t_2z$ bits, where t_1 is a security parameter and suggested to be 80 in practice (Kolesnikov et al., 2009). The CMP initialization also has some communication cost. We do not involve it, since it can be done before running the protocol in Step 3. The last step requires $12t_2$ bits. The communication cost of secure bit-length is thus $24t_2z + 14t_2 + 6(\lambda + z + 2)t_1z$ bits bounded by $O(z(t_2 + t_1\lambda))$.

Again, we prove the security of the bit-length protocol by simulation (Section 2.6.1).

Theorem 4.2 *In the secure bit-length protocol, Alice (Bob) only learns the protocol output to her (him).*

Proof 4.5 *We first simulate the view of Alice. Let a be the input of Alice, and (c_1, c_1') be the protocol output to her. Her view is $VIEW_1^{BL} = (a, V_1^1, V_1^2, V_1^3, V_1^4)$, where V_1^1 is $\forall_{i=1}^z(M_{i1}, M_{i2}, M_{i3}, M_{i4})$ generated in step 2, V_1^2 is $\forall_{i=1}^z(D[M_{i3}], D[M_{i4}])$ that are the decryptions of M_{i3} and M_{i4} in Step 3, V_1^3 is the set of messages she receives from Bob in Step 3 for the CMP comparison, and V_1^4 is the set of messages she receives for the*

(2,2)-threshold Paillier decryption in Step 4. The simulator $S_1^{BL}(a,(c_1,c_1'))$ *to simulate* $VIEW_1^{BL}$ *is created as follows. For each quadruple in* V_1^1*, it computes* $(E[r_1],E[r_2],E[r_3],E[r_4])$*, where* r_i*'s for* $1 \leq i \leq 4$ *are random values. By the property of semantic security of Paillier cryptosystem,* $(E[r_1],E[r_2],E[r_3],E[r_4])$ *is indistinguishable from the quadruple in* V_1^1*. To simulate* $D[M_{i3}]$ *(and* $D[M_{i4}]$*) in* V_1^2*, a random value* $r \in \{2^z, 2^z+1, \ldots, 2^z+2^\eta-1\}$ *is selected. According to Lemma 4.3,* $D[M_{i3}]$ *(and* $D[M_{i4}]$*) is statistically indistinguishable from* r*. The simulation of* V_1^3 *and* V_1^4 *are already given in (Kolesnikov and Schneider, 2008) and (Cramer et al., 2001), respectively.* S_1^{BL} *can call them for the simulation.*

Next, we simulate the view of Bob. Let b *be the input of Bob, and* (c_2,c_2') *be the protocol output to him. His view is* (b,V_2^1,V_2^2,V_2^3)*, where* V_2^1 *is quadruples generated in step 1 by Alice,* V_2^2 *is the set of messages he receives from Alice in Step 3 for the CMP comparison, and* V_2^3 *is the set of messages she receives for the (2,2)-threshold Paillier decryption in Step 4. The simulator* $S_2^{BL}(b,(c_2,c_2'))$ *to simulate* $VIEW_2^{BL}$ *is created as follows. To simulate a quadruple in* V_2^1*, it randomly selects* r_i*'s for* $1 \leq i \leq 4$ *and computes* $(E[r_1],E[r_2],E[r_3],E[r_4])$*, which is indistinguishable from the quadruple in* V_2^1 *by the property of Paillier cryptosystem. The simulation of* V_2^2 *and* V_2^3 *are already given in (Kolesnikov and Schneider, 2008) and (Cramer et al., 2001), respectively.* S_2^{BL} *can call them for the simulation.*

4.1.5 Secure Division

Let $A=(a_1,a_2)$ and $B=(b_1,b_2)$ be the respective inputs of Alice and Bob, where $a_2+b_2>0$. The secure division operation of $\frac{a_1+b_1}{a_2+b_2}$ is an SMC (secure multi-party computation) protocol between Alice and Bob. Alice and Bob first compute an encryption of I, where $I \approx \frac{2^\kappa}{a_2+b_2}$, and κ is a public parameter. Then, they compute c_1 and c_2, such that $c_1+c_2 = I(a_1+b_1)$, where Alice learns only c_1 and Bob learns only c_2. Finally, Alice and Bob divide c_1 and c_2 by 2^κ, and $\frac{c_1}{2^\kappa}+\frac{c_2}{2^\kappa}$ is the approximation of $\frac{a_1+b_1}{a_2+b_2}$.

The approximation of $\frac{2^\kappa}{a_2+b_2}$. Let $\alpha = \frac{a_2+b_2}{2^\ell}$, where $\ell = \lfloor \log(a_2+b_2) \rfloor + 1$. We approximate $\frac{2^k}{a_2+b_2}$ by Taylor series.

$$\frac{1}{\alpha} = \sum_{i=0}^{\infty}(1-\alpha)^i = \sum_{i=0}^{\omega}(1-\alpha)^i + \Delta_\omega, \tag{4.8}$$

where

$$\Delta_\omega = \sum_{i=\omega+1}^{\infty} (1-\alpha)^i \leq \frac{(1-\alpha)^{\omega+1}}{\alpha}. \tag{4.9}$$

It can be easily verified that

$$\frac{1}{2} \leq \left(\alpha = \frac{a_2+b_2}{2^\ell}\right) < 1. \tag{4.10}$$

Therefore, $\Delta_\omega \leq 2^{-\omega}$. By picking a sufficiently large ω, the additive error Δ_ω is appropriately small, so that it can be truncated in approximating $\frac{1}{\alpha}$.

We now plug the Taylor series of Equation 4.8 into $\frac{2^\kappa}{a_2+b_2}$. Thus,

$$\begin{aligned}
\frac{2^\kappa}{a_2+b_2} = \frac{2^{\kappa-\ell}}{\alpha} =& 2^{\kappa-\ell}\left(\sum_{i=0}^{\omega}(1-\alpha)^i\right) + 2^{\kappa-\ell}\Delta_\omega \\
=& 2^{\kappa-\ell(\omega+1)}\left(\sum_{i=0}^{\omega}\left(1-\frac{a_2+b_2}{2^\ell}\right)^i 2^{\ell\omega}\right) + 2^{\kappa-\ell}\Delta_\omega, \\
=& 2^{\kappa-\ell(\omega+1)}\left(\sum_{i=0}^{\omega}\left(1-\frac{a_2+b_2}{2^\ell}\right)^i (2^\ell)^i\, 2^{\ell(\omega-i)}\right) + 2^{\kappa-\ell}\Delta_\omega, \\
=& 2^{\kappa-\ell(\omega+1)}\left(\sum_{i=0}^{\omega}\left(2^\ell-(a_2+b_2)\right)^i 2^{\ell(\omega-i)}\right) + 2^{\kappa-\ell}\Delta_\omega,
\end{aligned} \tag{4.11}$$

where $\kappa - \ell(\omega+1) > 0$. We then approximate $\frac{2^\kappa}{a_2+b_2}$ by

$$\frac{2^\kappa}{a_2+b_2} \;\approx\; 2^{\kappa-\ell}\sum_{i=0}^{\omega}(1-\alpha)^i. \tag{4.12}$$

The Protocol of Secure Division. On the basis of the approximation of $\frac{2^\kappa}{a_2+b_2}$, we present the SMC protocol of secure division in Figure 4.3. Alice and Bob first configure a (2,2)-threshold Paillier cryptosystem. Let (pk, sk) be the public and private key pair of the cryptosystem. Suppose that sk_A and sk_B are the secret shares of Alice and Bob respectively, such that the shares combined together can recover sk. Let $E[.]$ and $D[.]$ be the encryption and decryption functions corresponding to pk, and (sk_A, sk_B), respectively. They carry out the protocol step by step as follows.

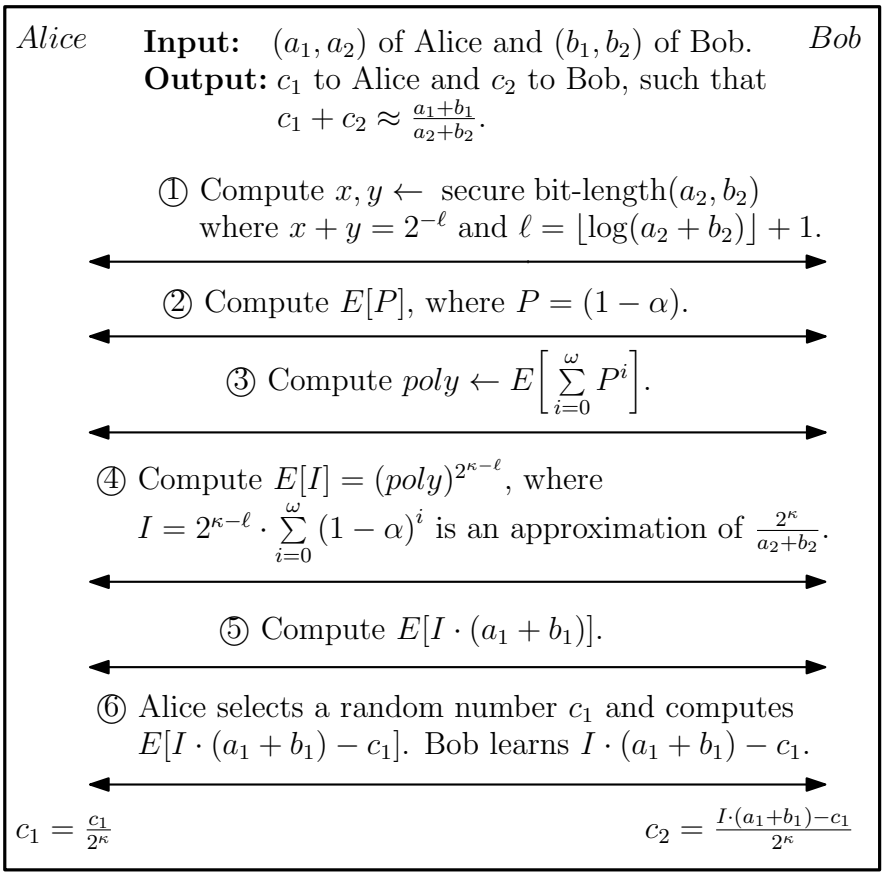


Figure 4.3: Secure division protocol

Step 1. Alice and Bob call *secure bit-length* operator (Section 4.1.4) to compute x and y, such that $x + y = 2^{-\ell} = 2^{-(\log\lfloor a_2+b_2\rfloor+1)} =$. Secure bit-length is a sub-protocol of the secure division. At the end of the sub-protocol, Alice learns only x, and Bob learns only y.

Step 2. Alice and Bob then jointly compute the encryption of $1 - \alpha$. Denote $1 - \alpha$ by P.

$$E[P] = E[1 - \alpha] = E[1 - \frac{a_2 + b_2}{2^\ell}] = E[1 - (a_2(x + y) + b_2(x + y))]$$
$$= \frac{E[1]}{(E[x] \times E[y])^{a_2} \times (E[x] \times E[y])^{b_2}}. \tag{4.13}$$

To compute $E[P]$, Alice and Bob jointly compute $E[x] \times E[y] = E[x + y]$. Alice then computes $(E[x] \times E[y])^{a_2} = E[a_2(x+y)]$. Bob also computes $(E[x] \times E[y])^{b_2} = E[b_2(x+y)]$.

Step 3. We use "square and multiply" approach to compute

$$poly = E\left[\sum_{i=0}^{\omega} P^i\right] = E[P^0] \times E[P^1] \times \cdots \times E[P^\omega], \tag{4.14}$$

where $P^0 = 1$ and $E[P^1]$ has been computed in Equation 4.13. $E[P^i]$ for $i \geq 2$ is computed by

$$E[P^i] = (E[P^{i-1}])^P = \frac{E[P^{i-1}]}{(E[P^{i-1}]^{a_2} \times E[P^{i-1}]^{b_2})^{2^{-\ell}}}. \tag{4.15}$$

The computation of $E[P^i]$ is similar to that of $E[P]$ in Equation 4.13.

Step 4. Let $I = 2^{\kappa-\ell} \cdot \sum_{i=0}^{\omega} (1-\alpha)^i$. Then, I is the approximation of $\frac{2^\kappa}{a_2+b_2}$ (Equation 4.12). Alice and Bob jointly compute

$$E[I] = (poly)^{2^{\kappa-\ell}} = (poly)^{2^\kappa (x+y)}, \tag{4.16}$$

which again can be computed similarly as $E[P]$ in Equation 4.13.

Step 5. Clearly, $I(a_1 + b_1)$ is an approximation of $\frac{2^\kappa(a_1+b_1)}{a_2+b_2}$. Alice computes $E[I]^{a_1}$ and Bob computes $E[I]^{b_1}$. It then follows that

$$E[I(a_1 + b_1)] = E[I]^{a_1} \times E[I]^{b_1}. \tag{4.17}$$

Step 6. Alice selects a random integer c_1 to compute

$$E[I(a_1 + b_1) - c_1] = \frac{E[I(a_1 + b_1)]}{E[c_1]}, \tag{4.18}$$

which is sent to Bob. Alice and Bob then jointly decrypt $E[I(a_1 + b_1) - c_1]$, and Bob learns $c_2 = I(a_1 + b_1) - c_1$. The statistical indistinguishability of c_1 and that of c_2 can be proven by using a similar proof as in Lemma 4.1.

At the end of the protocol, Alice and Bob divide c_1 and c_2 by 2^κ, respectively. It can be easily verified that

$$\frac{c_1}{2^\kappa} + \frac{c_2}{2^\kappa} \approx \frac{a_1 + b_1}{a_2 + b_2}. \tag{4.19}$$

4.1.5.1 The Protocol Analysis

We now analyze the protocol for its approximation error, complexity, and security. The lemma below gives the error bound when the inputs of secure division are integer.

Lemma 4.4 *Let a_1, a_2, b_1, b_2 be integers and $a_2 + b_2 > 0$. Then, the relative error of $\frac{a_1+b_1}{a_2+b_2}$ by the approximate Equation 4.19 is less than $(2)^{-\omega}$.*

Proof 4.6 *The relative error of* $\frac{a_1+b_1}{a_2+b_2}$ *in the secure division is*

$$\frac{\frac{2^\kappa(a_1+b_1)}{a_2+b_2} - (a_1+b_1)(2^{\kappa-\ell}\cdot\sum_{i=0}^{\omega}(1-\alpha)^i)}{\frac{2^\kappa(a_1+b_1)}{a_2+b_2}}. \tag{4.20}$$

By the Taylor series in Equation 4.11 and the approximation in Equation 4.12, it follows that

$$\frac{2^\kappa(a_1+b_1)}{a_2+b_2} - (a_1+b_1)\left(2^{\kappa-\ell}\cdot\sum_{i=0}^{\omega}(1-\alpha)^i\right) < 2^{\kappa-\ell}(a_1+b_1)\Delta_\omega. \tag{4.21}$$

Therefore, the relative error in Equation 4.20 is lower than

$$\frac{a_2+b_2}{2^\ell}\Delta_\omega = \alpha\Delta_\omega \leq \alpha(2)^{-\omega} < (2)^{-\omega} \tag{4.22}$$

where the first inequality holds since $\Delta_\omega \leq \frac{(1-\alpha)^{\omega+1}}{\alpha} \leq (2)^{-\omega}$ *(Inequality 4.9) and the second inequality holds since* $\frac{1}{2} \leq \alpha < 1$ *(Inequality 4.10).*

In the above lemma we consider that the inputs of the division operator are integers. Now we discuss the case that the inputs are floating values. We adopt the same strategy as that for secure multiplication to tranform floating values to integers.

Lemma 4.5 *Let* a_1, a_2, b_1, b_2 *be values larger than or equal to* $2^{-\gamma}$*. Suppose that each value is multiplied by* $\tau = 2^{\gamma+\beta+1}$ *and rounded to the integral part, before input to secure division operator. Then, the relative error of* $\frac{a_1+b_1}{a_2+b_2}$ *is less than* $\frac{1}{2^\beta} + \frac{2^{\beta-\omega}}{2^\beta-1}$*.*

Proof 4.7 *Let* $V = \frac{2^\kappa(a_1+b_1)}{(a_2+b_2)}$ *and* $V_A = \frac{2^\kappa(\lfloor a_1\times\tau\rfloor+\lfloor b_1\times\tau\rfloor)}{\lfloor a_2\times\tau\rfloor+\lfloor b_2\times\tau\rfloor}$*. Then, the approximation of* V_A *by secure division is*

$$V_B = 2^{\kappa-\ell'}(\lfloor a_1\times\tau\rfloor + \lfloor b_1\times\tau\rfloor)\left(\sum_{i=0}^{\omega}(1-\alpha')^i\right),$$

where $\alpha' = \frac{\lfloor a_2\times\tau\rfloor+\lfloor b_2\times\tau\rfloor}{2^{\ell'}}$ *and* $\ell' = \lfloor\log(\lfloor a_2\times\tau\rfloor + \lfloor b_2\times\tau\rfloor)\rfloor + 1$*. Then, the relative error is*

$$\frac{V-V_B}{V} = 1 - \frac{V_A}{V} + \frac{V_A}{V}\left(\frac{V_A-V_B}{V_A}\right). \tag{4.23}$$

We can easily verify that

$$1 - \frac{1}{2^\beta} \leq \frac{V_A}{V} \leq \frac{2^\beta}{2^\beta-1}. \tag{4.24}$$

Based on Lemma 4.4, the relative error of $\frac{V_A - V_B}{V_A}$ *is less than* $2^{-\omega}$*. Therefore, the relative error is*

$$\frac{V - V_B}{V} < \frac{1}{2^\beta} + \frac{2^{\beta-\omega}}{2^\beta - 1}, \tag{4.25}$$

which concludes the proof.

Time Complexity. We measure the time complexity by modular exponentiations, since they consume most of the time. Secure bit-length in Step 1 requires $36z+26$ modular exponentiations (refer to Section 4.1.4 for details). Step 3 iteratively computes the encryption of $\sum_{i=1}^{\omega} P^i$; it takes $4(\omega - 1)$ modular exponentiations. The other 4 steps together take 24 modular exponentiations. Therefore, the number of modular exponentiations needed by secure division is $4\omega + 46 + 36z$ bounded by $O(\omega + z)$.

Communication Complexity. We measure the communication complexity by the number of message bits passing between Alice and Bob. The communication cost of Step 1 is $24t_2z + 14t_2 + 6(\lambda + z + 2)t_1z$ bits (Section 4.1.4), where t_1 is a security parameter and t_2 is the message length in the Paillier cryptosystem (Note: t_1 is suggested to be 80 in practice (Kolesnikov et al., 2009)). The number of message bits passing between Alice and Bob in Step 3 is $8t_2(\omega - 1)$, and in the other 4 steps is $18t_2$. Therefore, the communication complexity is $t_2(24z + 24 + 8w) + 6(\lambda + z + 2)t_1z$ bits bounded by $O(z(t_2 + t_1\lambda) + t_2\omega)$.

Secure division is an SMC protocol. We prove its security by simulation methodology (Section 2.6.1 for more details).

Theorem 4.3 *The secure division protocol is simulatable.*

Proof 4.8 *We simulate the view of Alice and that of Bob. We first simulate the view of Alice. Let* (a_1, a_2) *be the inputs of Alice, and* c_1 *be the protocol output to her. According to the secure division protocol in Figure 4.3, the view of Alice is* $VIEW_1^{\text{div}} = ((a_1, a_2), \mathcal{V}_1^1, \mathcal{V}_1^2, \mathcal{V}_1^3)$*, where* $\mathcal{V}_1^1$ *is the set of messages she receives from Bob for the secure bit-length sub-protocol in Step 1,* $\mathcal{V}_1^2$ *is the set of encrypted messages she receives to compute* $E[I \cdot (a_1 + b_1)]$ *(in Steps 2, 3, 4, and 5), and* $\mathcal{V}_1^3$ *is the set of messages she receives for (2,2)-threshold Paillier decryption in Step 6. The simulator* $S_1^{\text{div}}((a_1, a_2), c_1)$ *to simulate* $VIEW_1^{\text{div}}$ *is created as follows.* S_1^{div} *first calls* $S_1^{\textit{bit-length}}$ *(see proof in Theorem 4.2) to simulate* $\mathcal{V}_1^1$*. Each message* $E[m] \in \mathcal{V}_1^2$ *is a (2,2)-threshold Paillier encryption. To simulate it,* S_1^{div} *selects a random value* r *and computes* $E[r]$*. Because (2,2)-threshold Paillier*

encryption is semantically secure, $E[m]$ and $E[r]$ are computationally indistinguishable. The simulation of $\mathcal{V}_1^3$ is already given in (Cramer et al., 2001). Thus, S_1^{div} can call the simulator in (Cramer et al., 2001) to simulate $\mathcal{V}_1^3$.

The view of Bob is simulated in the following. Let (b_1, b_2) be the inputs of Bob, and c_2 be the protocol output to him. According to the secure division protocol in Figure 4.3, the view of Bob is $VIEW_2^{\text{div}} = ((b_1, b_2), \mathcal{V}_2^1, \mathcal{V}_2^2, \mathcal{V}_2^3)$, where $\mathcal{V}_2^1$ is the set of messages he receives from Alice for the secure bit-length sub-protocol in Step 1, $\mathcal{V}_2^2$ is the set of encrypted messages he receives to compute $E[I \cdot (a_1 + b_1)]$ (in Steps 2, 3, 4, and 5), and $\mathcal{V}_2^3$ is the set of messages he receives for (2,2)-threshold Paillier decryption in Step 6. The simulator $S_2^{\text{div}}((b_1, b_2), c_2)$ to simulate $VIEW_2^{\text{div}}$ is created as follows. S_2^{div} first calls $S_2^{bit\text{-}length}$ (see proof in Theorem 4.2) to simulate $\mathcal{V}_2^1$. Each message $E[m_b] \in \mathcal{V}_2^2$ is a (2,2)-threshold Paillier encryption. To simulate it, S_2^{div} selects a random value r_b and computes $E[r_b]$. Because (2,2)-threshold Paillier encryption is semantically secure, $E[m_b]$ and $E[r_b]$ are computationally indistinguishable. The simulation of $\mathcal{V}_2^3$ is already given in (Cramer et al., 2001). Thus, S_2^{div} can call the simulator in (Cramer et al., 2001) to simulate $\mathcal{V}_2^3$.

4.1.6 Secure Log

The secure log operation is an SMC protocol to split $\ln(a + b)$ into two values that are distributed to Alice and Bob. Here a and b are the respective inputs of Alice and Bob and $a+b > 1$. On a high level, Alice and Bob first compute an encryption of $I \approx 2^\kappa \mathcal{F}_l \ln(a+b)$, where κ and $\mathcal{F}_l = \mathsf{LCM}(1, 2, \ldots, \omega)$ (i.e., the least common multiple) are public parameters. Then, they compute c_1 and c_2, such that $c_1 + c_2 = I$ and Alice (Bob) learns only c_1 (c_2). Finally, Alice and Bob divide c_1 and c_2 by $2^\kappa \mathcal{F}_l$, and $\frac{c_1}{2^\kappa \mathcal{F}_l} + \frac{c_2}{2^\kappa \mathcal{F}_l}$ is an approximation of $\ln(a + b)$. In the protocol we consider natural logarithm. Conversion to log with other base is trivial, e.g., on base 2, $\log(a + b) = \ln(a + b) \log(e)$.

The approximation of $2^\kappa \mathcal{F}_l \ln(\mathbf{a} + \mathbf{b})$. Again we use Taylor series to achieve the approximation. Let $\alpha = \frac{a+b}{2^\ell}$, where $\ell = \lfloor \log(a + b) \rfloor + 1$. The Taylor series of $\ln(\alpha)$ is as follows.

$$\ln(\alpha) = -\sum_{i=1}^{\infty} \frac{1}{i}(1 - \alpha)^i = -\sum_{i=1}^{\omega} \frac{1}{i}(1 - \alpha)^i - \Delta_\omega, \tag{4.26}$$

where

$$\Delta_\omega = \sum_{i=\omega+1}^{\infty} \frac{1}{i}(1-\alpha)^i < -\ln(\alpha)(1-\alpha)^\omega. \tag{4.27}$$

Since $\frac{1}{2} \leq \alpha < 1$, it follows that $\Delta_\omega < 2^{-\omega}\ln(2)$. Therefore, by picking a sufficiently large ω, the additive error Δ_ω is appropriately small and can be truncated in approximating $\ln(\alpha)$.

We now plug the Taylor series of Equation 4.26 into $2^k \cdot \mathcal{F}_l \cdot \ln(a+b)$. Thus,

$$\begin{aligned}
&2^\kappa \mathcal{F}_l \ln(a+b) = 2^\kappa \mathcal{F}_l \ln(\alpha) + 2^\kappa \mathcal{F}_l \ln(2^\ell) \\
&= 2^\kappa \mathcal{F}_l \left(-\sum_{i=1}^{\omega} \frac{(1-\alpha)^i}{i} \right) + 2^\kappa \mathcal{F}_l \ell \ln(2) - 2^\kappa \mathcal{F}_l \Delta_\omega \\
&= 2^{\kappa-\ell\omega} \left(-\sum_{i=1}^{\omega} \mathcal{F}_l \frac{(1-\alpha)^i}{i} \right) 2^{\ell\omega} + 2^\kappa \mathcal{F}_l \ell \ln(2) - 2^\kappa \mathcal{F}_l \Delta_\omega \\
&= 2^{\kappa-\ell\omega} \left(-\sum_{i=1}^{\omega} i^{-1} \mathcal{F}_l \left(1 - \frac{a+b}{2^\ell}\right)^i \right) (2^\ell)^i \, 2^{\ell(\omega-i)} + 2^\kappa \mathcal{F}_l \ell \ln(2) - 2^\kappa \mathcal{F}_l \Delta_\omega \\
&= 2^{\kappa-\ell\omega} \left(-\sum_{i=1}^{\omega} i^{-1} \mathcal{F}_l \left(2^\ell - (a+b)\right)^i 2^{\ell(\omega-i)} \right) + 2^\kappa \mathcal{F}_l \ell \ln(2) - 2^\kappa \mathcal{F}_l \Delta_\omega,
\end{aligned} \tag{4.28}$$

where $\kappa - \ell\omega > 0$. We thus obtain the approximation:

$$2^k \cdot \mathcal{F}_l \cdot \ln(a+b) \;\approx\; 2^\kappa \mathcal{F}_l \left(-\sum_{i=1}^{\omega} \frac{(1-\alpha)^i}{i} \right) + 2^\kappa \mathcal{F}_l \ell \ln(2). \tag{4.29}$$

The Protocol of Secure Log. On the basis of the approximation of $2^k \cdot \mathcal{F}_l \cdot \ln(a+b)$, we present the SMC protocol of secure log in Figure 4.4. Alice and Bob first configure a (2,2)-threshold Paillier cryptosystem (i.e. any cipher text can be decrypted only by Alice and Bob). Let (pk, sk) be the public and private key pair of the cryptosystem. Suppose that sk_A and sk_B are the secret shares of Alice and Bob respectively, such that the shares combined together can recover sk. Let $E[.]$ and $D[.]$ be the encryption and decryption functions corresponding to pk, and (sk_A, sk_B), respectively. Alice and Bob carry out the protocol step by step as follows.

Step 1. Alice and Bob call *secure bit-length* operator (Section 4.1.4) to compute x and x' (unknown only by Bob), and y and y' (unknown only by Alice), such that $x+y$ is equal to $2^{-(\lfloor \log(a+b) \rfloor + 1)}$ and $x' + y' = \lfloor \log(a+b) \rfloor + 1$.

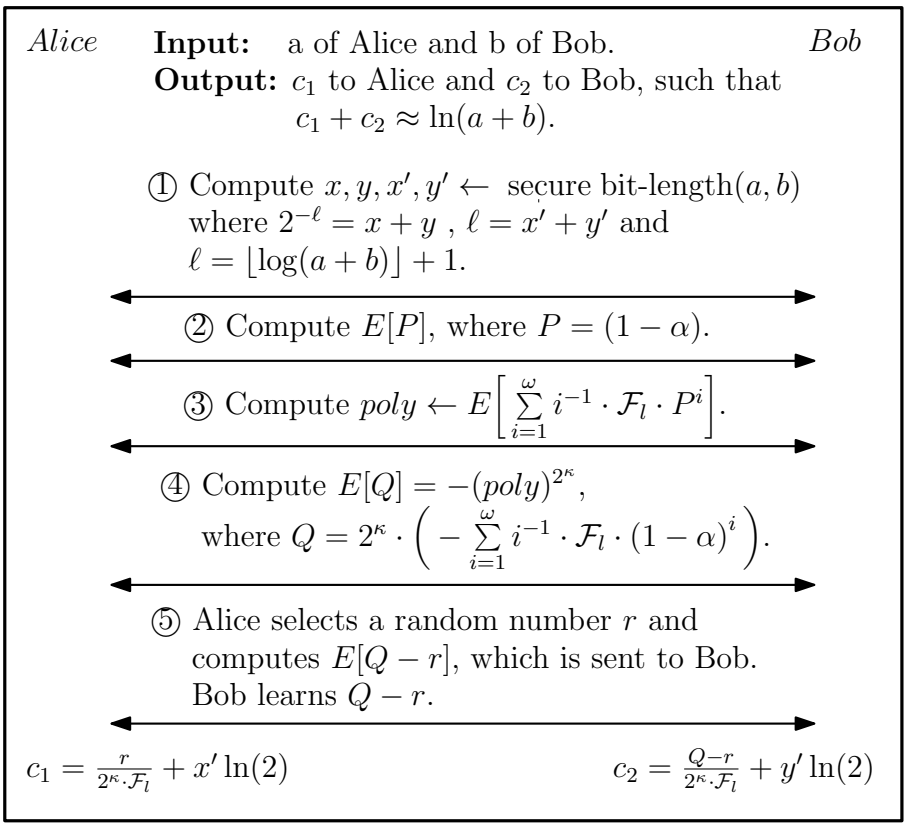


Figure 4.4: Secure log protocol

Step 2. Alice and Bob then jointly compute the encryption of $1-\alpha$. Denote $1-\alpha$ by P.

$$E[P] = E[1-\alpha] = E[1 - \frac{a+b}{2^\ell}] = E[1 - (a(x+y) + b(x+y))]$$
$$= \frac{E[1]}{(E[x] \times E[y])^a \times (E[x] \times E[y])^b}. \tag{4.30}$$

To compute $E[P]$, Alice and Bob jointly compute $E[x] \times E[y] = E[x+y]$. Alice then computes $(E[x] \times E[y])^a = E[a(x+y)]$. Bob also computes $(E[x] \times E[y])^b = E[b(x+y)]$.

Step 3. Alice and Bob use "square and multiply" approach to jointly compute

$$poly = E\left[\sum_{i=1}^{\omega} i^{-1} \mathcal{F}_l \cdot P^i\right] = E[P^1]^{\mathcal{F}_l} \times \cdots \times E[P^\omega]^{\omega^{-1}\mathcal{F}_l}. \tag{4.31}$$

Computing $E[P^i]$ from $E[P^{i-1}]$ for $i = 2, \ldots, \omega$ is the same as that in Equation 4.15.

Step 4. Alice or Bob computes $E[Q] = -(poly)^{2^\kappa}$, where

$$Q = 2^\kappa \mathcal{F}_l \left(-\sum_{i=1}^{\omega} i^{-1} \cdot (1-\alpha)^i\right). \tag{4.32}$$

Step 5. Alice selects a random integer r and computes $E[Q-r] = \frac{E[Q]}{E[r]}$, which is sent to Bob. The two parties then jointly decrypt $E[Q-r]$ and only Bob learns the decryption. Alice sets $c_1 = \frac{r}{2^\kappa \mathcal{F}_l} + x' \cdot \ln(2)$ and Bob sets $c_2 = \frac{Q-r}{2^\kappa \mathcal{F}_l} + y' \cdot \ln(2)$, where $\ell = x' + y'$. By approximate Equation 4.29 and Equation 4.32, it is easy to verify that

$$c_1 + c_2 = -\sum_{i=1}^{\omega} \frac{(1-\alpha)^i}{i} + \ell \ln(2), \tag{4.33}$$

which is an approximation of $\ln(a+b)$. The statistical indistinguishability of c_1 and that of c_2 can be proven by using a similar proof as in Lemma 4.1.

4.1.6.1 The Protocol Analysis

We now analyze the protocol for its approximation error, complexity, and security. The lemma below gives the error bound when the inputs of secure log are integer.

Lemma 4.6 *The relative error of* $\ln(a+b)$ *by the secure log operator is less than* $2^{-\omega}$*, when* a *and* b *are integers and* $a+b>1$.

Proof 4.9 *According to the Taylor series in Equation 4.28 and the approximation in Equation 4.29, the relative error is* $\frac{\Delta_\omega}{\ln(a+b)}$*, which combined with Inequality 4.27 gives*

$$\frac{\Delta_\omega}{\ln(a+b)} < \frac{-\ln(\alpha)(1-\alpha)^\omega}{\ln(\alpha)+\ell\ln(2)} \leq 2^{-\omega}, \tag{4.34}$$

where the second inequality holds since $1/2 \leq \alpha < 1$ *and* $\ell \geq 2$ *for* $a+b>1$.

Next we examine the error when a and b are floating values.

Lemma 4.7 *Let* a *and* b *be two values not less than* $2^{-\gamma}$*, and* $\tau = 2^{\gamma+\beta+1}$*. Both values are multiplied by* τ *and rounded to their integer parts, before input to the secure log operator. Then, the relative error of* $\ln(a+b)$ *is less than* $1 - \frac{\ln(2^{\beta+1}-2)}{\ln(2^{\beta+1})} + 2^{-\omega}$.

Proof 4.10 *Let* $V = 2^\kappa \mathcal{F}_l \ln(a\times\tau + b\times\tau)$ *and* $V_A = 2^\kappa \mathcal{F}_l \ln(\lfloor a\times\tau\rfloor + \lfloor b\times\tau\rfloor)$*. According to approximate Equation 4.29,* V_A *can be approximated by* V_B*:*

$$V_B = 2^\kappa \mathcal{F}_l \left(-\sum_{i=1}^{\omega} \frac{(1-\alpha')^i}{i} \right) + 2^\kappa \mathcal{F}_l \ell' \ln(2), \tag{4.35}$$

where $\alpha' = \frac{\lfloor a\times\tau\rfloor+\lfloor b\times\tau\rfloor}{2^{\ell'}}$ *and* $\ell' = \lfloor\log(\lfloor a\times\tau\rfloor + \lfloor b\times\tau\rfloor)\rfloor + 1$. *Then, the relative error is*

$$\frac{V - V_B}{V} = 1 - \frac{V_A}{V} + \frac{V_A}{V}\left(\frac{V_A - V_B}{V_A}\right). \tag{4.36}$$

Since $\lfloor a\times\tau\rfloor > 2^{\beta}$ *and* $\lfloor b\times\tau\rfloor > 2^{\beta}$, *it follows that*

$$\frac{\ln(2^{\beta+1}-2)}{\ln(2^{\beta+1})} < \frac{V_A}{V} \leq 1. \tag{4.37}$$

Based on Lemma 4.6, we have

$$\frac{V_A - V_B}{V_A} < 2^{-\omega} \tag{4.38}$$

Therefore, the relative error is

$$\frac{V - V_B}{V} < 1 - \frac{\ln(2^{\beta+1}-2)}{\ln(2^{\beta+1})} + 2^{-\omega}. \tag{4.39}$$

Time Complexity. We measure the time complexity by modular exponentiations, since they consume most of the time. Secure bit-length in Step 1 requires $36z + 26$ modular exponentiations. Step 3 iteratively computes the encryption of $\sum_{i=1}^{\omega} i^{-1}\cdot\mathcal{F}\cdot P^i$; it takes $5(\omega-1)+1$ modular exponentiations. The other 3 steps together take 21 modular exponentiations. Therefore, the number of modular exponentiations needed by secure log is $5\omega + 43 + 36z$ bounded by $O(\omega + z)$.

Communication Complexity. We measure the communication complexity by the number of message bits passing between Alice and Bob. The communication cost of Step 1 is $24t_2z + 14t_2 + 6(\lambda + z + 2)t_1z$ bits (Section 4.1.4), where t_1 is a security parameter and t_2 is the message length in the Paillier cryptosystem (Note: t_1 is suggested to be 80 in practice (Kolesnikov et al., 2009)). The number of message bits passing between Alice and Bob in Step 3 is $8t_2(\omega-1)$, and in other 3 steps it is $12t_2$, where t_2 is the message length in Paillier cryptosystem. Therefore, the communication complexity is $t_2(24z + 18 + 8w) + 6(\lambda + z + 2)t_1z$ bits bounded by $O(z(t_2 + t_1\lambda) + t_2\omega)$.

Secure log is proven secure via simulation paradigm (refer to Section 2.6.1 for more details) in the following.

Theorem 4.4 *The secure log protocol is simulatable.*

Proof 4.11 *We simulate the view of Alice and that of Bob. Let a be the input of Alice, and c_1 be the protocol output to her. According to the secure log protocol in Figure 4.4, the view of Alice is $VIEW_1^{\log} = (a, \mathcal{V}_1^1, \mathcal{V}_1^2, \mathcal{V}_1^3)$, where $\mathcal{V}_1^1$ is the set of messages she receives from Bob for the secure bit-length sub-protocol in Step 1, $\mathcal{V}_1^2$ is the set of encrypted messages she receives to compute $E[2^\kappa \cdot (-\sum_{i=1}^{\omega} i^{-1} \cdot \mathcal{F} \cdot (1-\alpha)^i)]$ (in Steps 2, 3, and 4), and $\mathcal{V}_1^3$ is the set of messages she receives for (2,2)-threshold Paillier decryption in Step 5. The simulator $S_1^{\log}(a, c_1)$ to simulate $VIEW_1^{\log}$ is created as follows. $S_1^{\log}$ first calls $S_1^{bit\text{-}length}$ (see proof in Theorem 4.2) to simulate $\mathcal{V}_1^1$. Each message $E[m] \in \mathcal{V}_1^2$ is a (2,2)-threshold Paillier encryption. To simulate it, $S_1^{\log}$ selects a random value r and computes $E[r]$. Because (2,2)-threshold Paillier encryption is semantically secure, $E[m]$ and $E[r]$ are computationally indistinguishable. The simulation of $\mathcal{V}_1^3$ is already given in (Cramer et al., 2001). Thus, S_1^{log} can call the simulator in (Cramer et al., 2001) to simulate $\mathcal{V}_1^3$.*

The view of Bob is in the following. Let b be the input of Bob, and c_2 be the protocol output to him. According to the secure log protocol in Figure 4.4, the view of Bob is $VIEW_2^{\log} = (b, \mathcal{V}_2^1, \mathcal{V}_2^2, \mathcal{V}_2^3)$, where $\mathcal{V}_2^1$ is the set of messages he receives from Alice for the secure bit-length sub-protocol in Step 1, $\mathcal{V}_2^2$ is the set of encrypted messages he receives to compute $E[2^\kappa \cdot (-\sum_{i=1}^{\omega} i^{-1} \cdot \mathcal{F} \cdot (1-\alpha)^i)]$ (in Steps 2, 3, and 4), and $\mathcal{V}_2^3$ is the set of messages she receives for (2,2)-threshold Paillier decryption in Step 5. The simulator $S_2^{\log}(b, c_2)$ to simulate $VIEW_2^{\log}$ is created as follows. $S_2^{\log}$ first calls $S_2^{bit\text{-}length}$ (see proof in Theorem 4.2) to simulate $\mathcal{V}_2^1$. Each message $E[m_b] \in \mathcal{V}_2^2$ is a (2,2)-threshold Paillier encryption. To simulate it, $S_2^{\log}$ selects a random value r_b and computes $E[r_b]$. Because (2,2)-threshold Paillier encryption is semantically secure, $E[m_b]$ and $E[r_b]$ are computationally indistinguishable. The simulation of $\mathcal{V}_2^3$ is already given in (Cramer et al., 2001). Thus, S_2^{log} can call the simulator in (Cramer et al., 2001) to simulate $\mathcal{V}_2^3$.

4.1.7 Secure Power

The secure power operator is an SMC protocol decomposing $(a+b)^{\frac{m}{d}}$ into c_1 (known only to Alice) and c_2 (known only to Bob), where a and b are the respective inputs of Alice and Bob, and m and d are positive integers with $m < d$. In the protocol, the two parties first compute an encryption of I, where $I \approx 2^\kappa \mathcal{F}_p (a+b)^{\frac{m}{d}}$ is obtained by Taylor series, κ is a public parameter, and $\mathcal{F}_p = \mathsf{LCM}(1, 2, \ldots, \omega) \cdot d^\omega$ (LCM is the least common multiple and ω is a threshold). They then compute c_1 and c_2 satisfying $c_1 + c_2 = I$. Finally, they

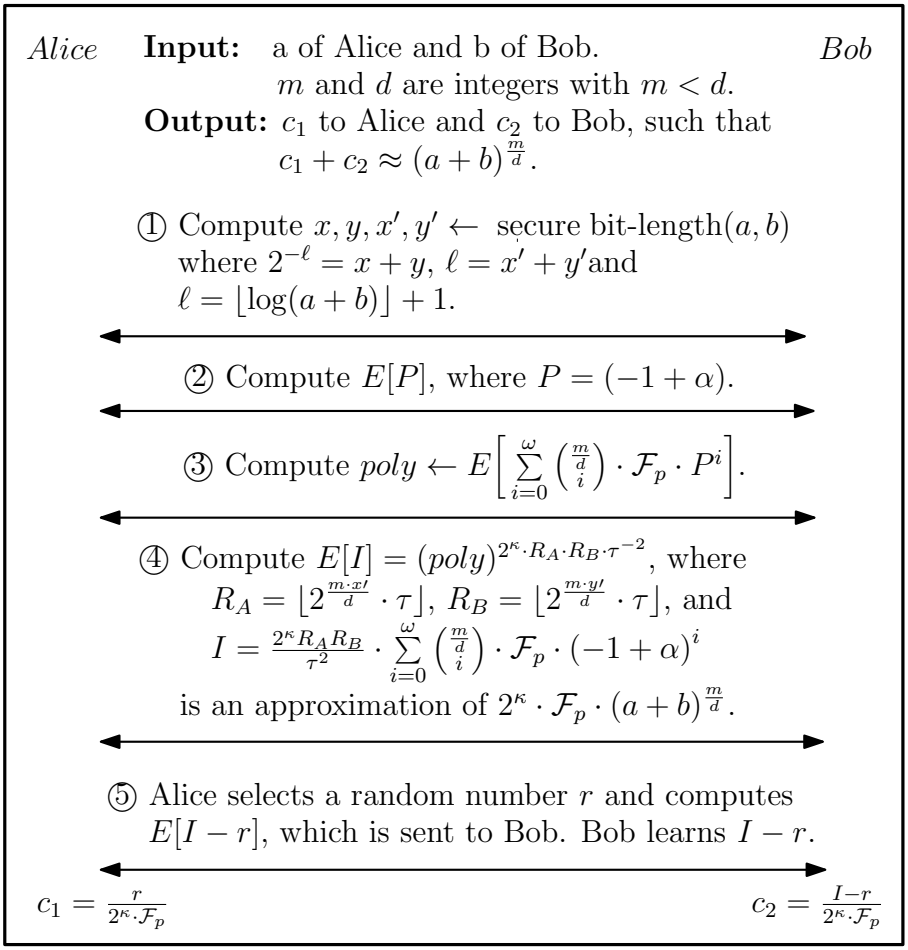


Figure 4.5: Secure power $\frac{m}{n}$ protocol

divide c_1 and c_2 by $2^\kappa \mathcal{F}_p$, and $\frac{c_1}{2^\kappa \mathcal{F}_p} + \frac{c_2}{2^\kappa \mathcal{F}_p}$ is an approximation of $(a+b)^{\frac{m}{d}}$. We derive $(x)^{\frac{m}{d}}$ in Taylor Series in the following.

Derivation of $(x)^{\frac{m}{d}}$ in Taylor Series. Let function $f(x)$ be centered at the point v, where derivatives $f^0(v), f^1(v), f^2(v), \cdots, f^\omega(v)$ all exist. We derive the Taylor Polynomial of $f(x)$ in the following.

$$P_\omega(x) = \sum_{i=0}^{\omega} \frac{f^{(i)}(v)(x-v)^i}{i!} = f^0(v) + f^1(v)(x-v) + \frac{f^2(v)(x-v)^2}{2!} + \frac{f^3(v)(x-v)^3}{3!} + \cdots + \frac{f^\omega(v)(x-v)^\omega}{\omega!} \tag{4.40}$$

We next find $f^0(x), f^1(x), f^2(x), \cdots, f^\omega(x)$ for $f(x) = (x)^{\frac{m}{d}}$ centered at 1 (i.e., $x = 1$),

$$
\begin{aligned}
f^0(x) &= (x)^{\frac{m}{d}} \Rightarrow f(1) = 1 \\
f^1(x) &= \left(\frac{m}{d}\right)(x)^{\frac{m}{d}-1} \Rightarrow f^1(1) = \frac{m}{d} \\
f^2(x) &= \left(\frac{m}{d}\right)\left(\frac{m}{d}-1\right)(x)^{\frac{m}{d}-2} \Rightarrow f^2(1) = \left(\frac{m}{d}\right)\left(\frac{m}{d}-1\right) \\
&\vdots \\
f^\omega(x) &= \left(\frac{m}{d}\right)\left(\frac{m}{d}-1\right)\cdots\left(\frac{m}{d}+1-\omega\right)(x)^{\frac{m}{d}-\omega} \\
&\Rightarrow f^\omega(1) = \left(\frac{m}{d}\right)\left(\frac{m}{d}-1\right)\cdots\left(\frac{m}{d}+1-\omega\right)
\end{aligned}
$$

The function $(x)^{\frac{m}{d}}$ can be presented in Taylor series by applying $f^0(x), f^1(x), f^2(x), \cdots, f^\omega(x)$ above into Equation 4.40 as follows.

$$
\begin{aligned}
&(x)^{\frac{m}{d}} = f^0(1) + f^1(1)(x-1) + \frac{f^2(1)(x-1)^2}{2!} + \frac{f^3(1)(x-1)^3}{3!} + \cdots + \frac{f^\omega(1)(x-1)^\omega}{\omega!} \\
&= 1 + \left(\frac{m}{d}\right)(x-1) + \frac{\left(\frac{m}{d}\right)\left(\frac{m}{d}-1\right)(x-1)^2}{2!} + \cdots + \frac{\left(\frac{m}{d}\right)\left(\frac{m}{d}-1\right)\cdots\left(\frac{m}{d}+1-\omega\right)(x-1)^\omega}{\omega!} \\
&= \sum_{i=0}^{\omega}\binom{\frac{m}{d}}{i}(-1+x)^i, \qquad (4.41)
\end{aligned}
$$

where m and d are integers, and $m < d$. Thus, $(x)^{\frac{m}{d}}$ is approximately equivalent to $\sum_{i=0}^{\omega}\binom{\frac{m}{d}}{i}(-1+x)^i$ as depicted in Equation 4.41.

The approximation of $2^\kappa \mathcal{F}_p(a+b)^{\frac{m}{d}}$. Let $\alpha = \frac{a+b}{2^\ell}$, where $\ell = \lfloor \log(a+b) \rfloor + 1$. The approximation of $(\alpha)^{\frac{m}{d}}$ by Taylor series is:

$$
(\alpha)^{\frac{m}{d}} = \sum_{i=0}^{\omega}\binom{\frac{m}{d}}{i}(-1+\alpha)^i + \Delta_\omega, \qquad (4.42)
$$

where

$$
\Delta_\omega = \sum_{i=\omega+1}^{\infty}\binom{\frac{m}{d}}{i}(-1+\alpha)^i < \left|(-1+\alpha)^{\omega+1}(\alpha)^{\frac{m}{d}}\right|. \qquad (4.43)
$$

The setting of ℓ gives that $0 < 1-\alpha \leq 1/2$. Plugging it into Inequality 4.43, it gives that

$$
\Delta_\omega < \left(\frac{1}{2}\right)^{(\omega+\frac{m}{d}+1)} \qquad (4.44)
$$

Therefore, setting ω to a sufficiently large value, Δ_ω is negligible and it can be truncated in approximating $(\alpha)^{\frac{m}{d}}$.

We now plug the Taylor series of Equation 4.42 into $2^\kappa \cdot \mathcal{F}_p \cdot (a+b)^{\frac{m}{d}}$. Thus,

$$\begin{aligned}
&2^\kappa \mathcal{F}_p (a+b)^{\frac{m}{d}} = 2^{\kappa+\frac{\ell m}{d}} \mathcal{F}_p (\alpha)^{\frac{m}{d}} \\
=&2^{\kappa+\frac{m\ell}{d}} \mathcal{F}_p \left(\sum_{i=0}^{\omega} \binom{\frac{m}{d}}{i} (-1+\alpha)^i \right) + 2^{\kappa+\frac{m\ell}{d}} \mathcal{F}_p \Delta_\omega \\
=&2^{\kappa+\frac{m\ell}{d}-\ell\omega} \left(\sum_{i=0}^{\omega} \binom{\frac{m}{d}}{i} \mathcal{F}_p (-1+\alpha)^i \ 2^{\ell\omega} \right) + 2^{\kappa+\frac{m\ell}{d}} \mathcal{F}_p \Delta_\omega \\
=&2^{\kappa+\frac{m\ell}{d}-\ell\omega} \left(\sum_{i=0}^{\omega} \binom{\frac{m}{d}}{i} \mathcal{F}_p (-1+\alpha)^i \ (2^\ell)^i \ 2^{\ell(\omega-i)} \right) + 2^{\kappa+\frac{m\ell}{d}} \mathcal{F}_p \Delta_\omega \\
=&2^{\kappa+\ell(\frac{m}{d}-\omega)} \left(\sum_{i=0}^{\omega} \binom{\frac{m}{d}}{i} \mathcal{F}_p \left(-2^\ell + (a+b)\right)^i 2^{\ell(\omega-i)} \right) + 2^{\kappa+\frac{m\ell}{d}} \mathcal{F}_p \Delta_\omega,
\end{aligned} \tag{4.45}$$

where $\kappa + \ell(\frac{m}{d} - \omega) > 0$. Consequently, we have

$$2^\kappa \cdot \mathcal{F}_p \cdot (a+b)^{\frac{m}{d}} \ \approx \ 2^{\kappa+\frac{m\ell}{d}} \mathcal{F}_p \left(\sum_{i=0}^{\omega} \binom{\frac{m}{d}}{i} (-1+\alpha)^i \right). \tag{4.46}$$

Note that in the above approximation, $2^{\frac{m\ell}{d}}$ may not be an integer. From $\ell = x' + y'$, where both x' and y' are integers, we set $R_A = \lfloor 2^{\frac{m \cdot x'}{d}} \cdot \tau \rfloor$ and $R_B = \lfloor 2^{\frac{m \cdot y'}{d}} \cdot \tau \rfloor$. As such, we further approximate

$$2^\kappa \cdot \mathcal{F}_p \cdot (a+b)^{\frac{m}{d}} \approx I, \tag{4.47}$$

where

$$I = \frac{2^\kappa R_A R_B}{\tau^2} \mathcal{F}_p \left(\sum_{i=0}^{\omega} \binom{\frac{m}{d}}{i} (-1+\alpha)^i \right). \tag{4.48}$$

The Protocol of Secure Power. On the basis of the approximation of $2^\kappa \cdot \mathcal{F}_p \cdot (a+b)^{\frac{m}{d}}$, we present the SMC protocol of secure power $\frac{m}{d}$ in Figure 4.5. Alice and Bob first configure a (2,2)-threshold Paillier cryptosystem (i.e. any cipher text can be decrypted only by Alice and Bob). Let (pk, sk) be the public and private key pair of the cryptosystem. Suppose that sk_A and sk_B are the secret shares of Alice and Bob respectively, such that the shares combined together can recover sk. Let $E[.]$ and $D[.]$ be the encryption and decryption functions corresponding to pk and (sk_A, sk_B), respectively. Alice and Bob carry out the protocol step by step as follows.

Step 1. Alice and Bob call *secure bit-length* operator (Section 4.1.4) to securely learn x and x' (known only to Alice), and y and y' (known only to Bob), such that $x + y = 2^{-\ell}$, $x' + y' = \ell$, and $\ell = \lfloor \log(a + b) \rfloor + 1$.

Step 2. Alice and Bob then jointly compute the encryption of $P = -1 + \alpha$. Denote $-1 + \alpha$ by P.

$$\begin{aligned} E[P] &= E[-1 + \alpha] = E[-1 + \frac{a+b}{2^{\ell}}] = E[-1 + (a(x+y) + b(x+y))] \\ &= \frac{(E[x] \times E[y])^a \times (E[x] \times E[y])^b}{E[1]}. \end{aligned} \tag{4.49}$$

To compute $E[P]$, Alice and Bob jointly compute $E[x] \times E[y] = E[x + y]$. Alice then computes $(E[x] \times E[y])^a = E[a(x + y)]$. Bob also computes $(E[x] \times E[y])^b = E[b(x + y)]$.

Step 3. Alice and Bob apply "square and multiply" approach to jointly compute

$$\begin{aligned} poly &= E\Big[\sum_{i=0}^{\omega} \binom{\frac{m}{d}}{i} \mathcal{F}_p \cdot P^i\Big] \\ &= E[P^0]^{\mathcal{F}_p} \times E[P^1]^{\frac{m}{d}\mathcal{F}_p} \times \cdots \times E[P^{\omega}]^{\binom{\frac{m}{d}}{\omega}\mathcal{F}_p}, \end{aligned}$$

where the computation of $E[P^i]$ from $E[P^{i-1}]$ is similar to Equation 4.15 for $i = 2, 3, \ldots, \omega$.

Step 4. Alice and Bob compute

$$E[I] = \Big(\Big((poly)^{2^{\kappa}\tau^{-2}}\Big)^{R_A}\Big)^{R_B}, \tag{4.50}$$

where $R_A = \lfloor 2^{\frac{m \cdot x'}{d}} \cdot \tau \rfloor$ and $R_B = \lfloor 2^{\frac{m \cdot y'}{d}} \cdot \tau \rfloor$.

Step 5. Alice selects a random integer r to compute $E[I - r] = \frac{E[I]}{E[r]}$, which is sent to Bob. They jointly decrypt the ciphertext, and only Bob learns $I - r$. Alice sets $c_1 = \frac{r}{2^{\kappa}\mathcal{F}_p}$, and Bob sets $c_2 = \frac{I-r}{2^{\kappa}\mathcal{F}_p}$. Clearly, $c_1 + c_2$ is an approximation of $(a + b)^{\frac{m}{d}}$ (by approximate Equation 4.47 and Equation 4.48). The statistical indistinguishability of c_1 and that of c_2 can be proven by using a similar proof as in Lemma 4.1.

4.1.7.1 The Protocol Analysis

We now analyze the protocol on the approximation error, complexity, and security. We first study the error, when a and b are integers and $a + b > 0$.

Lemma 4.8 *Let a and b be integers and $a+b>0$. Then, the relative error of $(a+b)^{\frac{m}{d}}$ by secure power is less than $2^{-\beta}+(1+2^{-\beta})2^{-(\omega+1)}$.*

Proof 4.12 *Let $V=2^{\kappa}\mathcal{F}_p(a+b)^{\frac{m}{d}}$. Then its approximation by secure power is I (approximate Equation 4.47). The relative error is*

$$\frac{|V-I|}{V}\leq\frac{|V-V_A|}{V}+\frac{V_A}{V}\left(\frac{V_A-I}{V_A}\right), \tag{4.51}$$

where V_A is another approximation of V by Equation 4.46.

$$V_A=2^{\kappa+\frac{m\ell}{d}}\mathcal{F}_p\sum_{i=0}^{\omega}\binom{\frac{m}{d}}{i}(-1+\alpha)^i. \tag{4.52}$$

By Equations 4.43, 4.45 and 4.46, it follows that

$$\frac{|V-V_A|}{V}\leq 2^{-(\omega+1)}\ \textit{and}\ \frac{V_A}{V}\leq 1+2^{-(\omega+1)} \tag{4.53}$$

Based on Lemma 4.2, we have

$$\frac{V_A-I}{V_A}<2^{-\beta}. \tag{4.54}$$

Therefore, the relative error is less than $2^{-\beta}+(1+2^{-\beta})2^{-(\omega+1)}$.

Like the error analysis for secure division and secure log, we have also the error analysis for secure power, when a and b are floating values.

Lemma 4.9 *Let a and b be two values not less than $2^{-\gamma}$, and $\tau=2^{\gamma+\beta+1}$. Both values are multiplied by τ and rounded to their integer parts, before input to secure power. Then, the relative error of $(a+b)^{\frac{m}{d}}$ is less than $1-\left(1-\frac{1}{2^{\beta}}\right)^{\frac{m}{d}}+2^{-(\omega+1)}+2^{-\beta}$.*

Proof 4.13 *Let $V=2^{\kappa}\mathcal{F}_p(a\times\tau+b\times\tau)^{\frac{m}{d}}$, and $V_A=2^{\kappa}\mathcal{F}_p(\lfloor a\times\tau\rfloor+\lfloor b\times\tau\rfloor)^{\frac{m}{d}}$. According to approximate Equation 4.47 and Equation 4.48, we can V by I':*

$$I'=\frac{2^{\kappa}R'_AR'_B}{\tau^2}\cdot\mathcal{F}_p\sum_{i=0}^{\omega}\binom{\frac{m}{d}}{i}(-1+\alpha')^i, \tag{4.55}$$

where $\alpha'=\frac{\lfloor a\times\tau\rfloor+\lfloor b\times\tau\rfloor}{2^{\ell'}}$, $\ell'=\lfloor\log(\lfloor a\times\tau\rfloor+\lfloor b\times\tau\rfloor)\rfloor+1$, $R'_A=\lfloor 2^{\frac{m\cdot x''}{d}}\cdot\tau\rfloor$, $R'_B=\lfloor 2^{\frac{m\cdot y''}{d}}\cdot\tau\rfloor$, and $x''+y''=\ell'$. Then, the relative error is

$$\frac{V-I'}{V}=1-\frac{V_A}{V}+\frac{V_A}{V}\frac{(V_A-I')}{V_A}. \tag{4.56}$$

Since $\lfloor a \times \tau \rfloor > 2^{\beta}$ *and* $\lfloor b \times \tau \rfloor > 2^{\beta}$*, it follows that*

$$\left(1 - \frac{1}{2^{\beta}}\right)^{\frac{m}{d}} \leq \frac{V_A}{V} \leq 1. \tag{4.57}$$

Based on Lemma 4.8,

$$\frac{V_A - I'}{V_A} < 2^{-\beta} + (1 + 2^{-\beta})2^{-(\omega+1)}. \tag{4.58}$$

Therefore, the relative error is bounded by

$$\frac{V - I'}{V} < 1 - \left(1 - \frac{1}{2^{\beta}}\right)^{\frac{m}{d}} 2^{-\beta} + (1 + 2^{-\beta})2^{-(\omega+1)}.$$

Time Complexity. We measure the time complexity by modular exponentiations, since they consume most of time. Secure bit-length in Step 1 requires $36z+26$ modular exponentiations. Step 3 iteratively computes the encryption of $\sum_{i=0}^{w} \binom{\frac{m}{d}}{i} P^i$; it takes $5(\omega - 1) + 2$ modular exponentiations. The other 3 steps together take 22 modular exponentiations. Therefore, the number of modular exponentiations needed by secure power $(a + b)^{\frac{m}{d}}$ is $5\omega + 45 + 36z$ bounded by $O(\omega + z)$.

Communication Complexity. We measure the communication complexity by the number of message bits passing between Alice and Bob. The communication cost of Step 1 is $24t_2z+14t_2+6(\lambda+z+2)t_1z$ bits (Section 4.1.4), where t_1 is a security parameter and t_2 is the message length in the Paillier cryptosystem (Note: t_1 is suggested to be 80 in practice (Kolesnikov et al., 2009)). The number of message bits passing between Alice and Bob in Step 3 is $8t_2(\omega-1)$, and in the other 3 steps it is $14t_2$. Therefore, the communication cost is $t_2(24z + 20 + 8w) + 6(\lambda + z + 2)t_1z$ bits bounded by $O(z(t_2 + t_1\lambda) + t_2\omega)$.

Next, we prove the secure power protocol via simulation paradigm (refer to Section 2.6.1 for more details) in the following.

Theorem 4.5 *The secure power protocol is simulatable.*

Proof 4.14 *We simulate the view of Alice and that of Bob. Let a be the input of Alice, and* c_1 *be the protocol output to her. According to the secure power protocol in Figure 4.5, the view of Alice is* $VIEW_1^{\frac{\mathrm{m}}{\mathrm{n}}} = (a, \mathcal{V}_1^1, \mathcal{V}_1^2, \mathcal{V}_1^3)$*, where* $\mathcal{V}_1^1$ *is the set of messages she receives from Bob for the secure bit-length sub-protocol in Step 1,* $\mathcal{V}_1^2$ *is the set of encrypted messages she receives to compute* $E[\frac{2^{\kappa}R_AR_B}{\tau^2} \cdot (\sum_{i=0}^{\omega} \binom{\frac{m}{n}}{i} \cdot \mathcal{F}_p \cdot (-1+\alpha)^i)]$ *(in Steps 2, 3,*

and 4), and $\mathcal{V}_1^3$ is the set of messages she receives for (2,2)-threshold Paillier decryption in Step 5. The simulator $S_1^{\frac{m}{n}}(a, c_1)$ to simulate $VIEW_1^{\frac{m}{n}}$ is created as follows. $S_1^{\frac{m}{n}}$ first calls $S_1^{bit\text{-}length}$ (see proof in Theorem 4.2) to simulate $\mathcal{V}_1^1$. Each message $E[m] \in \mathcal{V}_1^2$ is a (2,2)-threshold Paillier encryption. To simulate it, $S_1^{\frac{m}{n}}$ selects a random value r and computes $E[r]$. Because (2,2)-threshold Paillier encryption is semantically secure, $E[m]$ and $E[r]$ are computationally indistinguishable. The simulation of $\mathcal{V}_1^3$ is already given in (Cramer et al., 2001). Thus, $S_1^{\frac{m}{n}}$ can call the simulator in (Cramer et al., 2001) to simulate $\mathcal{V}_1^3$.

The view of Bob is simulated in the following.Let b be the input of Bob, and c_2 be the protocol output to him. According to the secure power protocol in Figure 4.5, the view of Bob is $VIEW_2^{\frac{m}{n}} = (b, \mathcal{V}_2^1, \mathcal{V}_2^2, \mathcal{V}_2^3)$, where $\mathcal{V}_2^1$ is the set of messages he receives from Alice for the secure bit-length sub-protocol in Step 1, $\mathcal{V}_2^2$ is the set of encrypted messages he receives to compute $E[\frac{2^\kappa R_A R_B}{\tau^2} \cdot (\sum_{i=0}^{\omega} \binom{m}{i} \cdot \mathcal{F}_p \cdot (-1+\alpha)^i)]$ (in Steps 2, 3, and 4), and $\mathcal{V}_2^3$ is the set of messages he receives for (2,2)-threshold Paillier decryption in Step 5. The simulator $S_2^{\frac{m}{n}}(b, c_2)$ to simulate $VIEW_2^{\frac{m}{n}}$ is created as follows. $S_2^{\frac{m}{n}}$ first calls $S_2^{bit\text{-}length}$ (see proof in Theorem 4.2) to simulate $\mathcal{V}_2^1$. Each message $E[m_b] \in \mathcal{V}_2^2$ is a (2,2)-threshold Paillier encryption. To simulate it, $S_2^{\frac{m}{n}}$ selects a random value r_b and computes $E[r_b]$. Because (2,2)-threshold Paillier encryption is semantically secure, $E[m_b]$ and $E[r_b]$ are computationally indistinguishable. The simulation of $\mathcal{V}_2^3$ is already given in (Cramer et al., 2001). Thus, $S_2^{\frac{m}{n}}$ can call the simulator in (Cramer et al., 2001) to simulate $\mathcal{V}_2^3$.

4.1.8 Secure Max

Figure 4.6 gives the SMC protocol of secure max. Let a and b be the private inputs of Alice and Bob, respectively. Alice and Bob compute $E[a]$ and $E[b]$, respectively, and input them to the secure max operator. The protocol outputs c_1 and c_2, such that $c_1 + c_2 = \max\{a, b\}$. In the protocol, a is kept confidential to Bob, and b is confidential to Alice. At the end of the protocol, Alice learns only c_1, and Bob learns only c_2.

Alice and Bob first configure a (2,2)-threshold Paillier cryptosystem. Let (pk, sk) be the public and private key pair of the cryptosystem. Suppose that sk_A and sk_B are the secret shares of Alice and Bob respectively, such that the shares combined together can recover sk. Let $E[.]$ and $D[.]$ be the encryption and decryption functions corresponding to pk, and (sk_A, sk_B), respectively. Alice and Bob carry out the protocol step by step as follows.

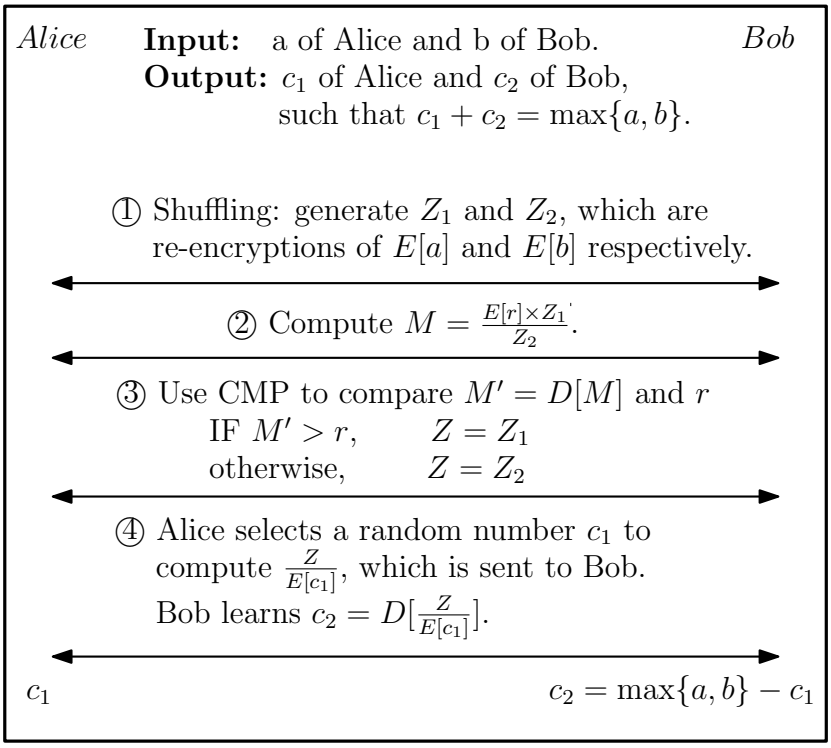


Figure 4.6: Secure max protocol

Step 1. Alice and Bob shuffle $E[a]$ and $E[b]$ as follow. Alice first sends $E[a]$ to Bob. Then, Bob re-encrypts a by $U_1 = E[a + 0] = E[a] \times E[0]$, and computes $U_2 = E[b]$. He sends (U_1, U_2) to Alice. Because of the semantic security of Paillier encryption, Alice cannot distinguish U_1 from U_2. She further re-encrypts a and b by

$$Z_1 = U_1 \times E[0],$$

$$Z_2 = U_2 \times E[0].$$

Based on the semantic security of Paillier encryption, neither Alice nor Bob can tell Z_1 (or Z_2) is the encryption of a or b.

Step 2. Alice computes $\frac{Z_1}{Z_2}$ which is sent to Bob. Bob then generates a random number r to compute M,

$$M = \frac{E[r] \times Z_1}{Z_2}, \tag{4.59}$$

which is sent to Alice.

Step 3. Alice receives M from Bob. They jointly decrypt $D[M] = M'$ which M' is known by Alice only. Alice and Bob then apply CMP (i.e., the secure integer comparison circuit in Section 3.1.10) to compare between M' and r which r is known by Bob only. Obviously, if $M' > r$, then Z_1 is selected, and Z_2 otherwise. Alice and Bob thus set

$$Z = \begin{cases} Z_1 & \text{if } M' > r, \\ Z_2 & \text{otherwise.} \end{cases} \tag{4.60}$$

Then, $Z = E[\max\{a, b\}]$.

Step 4. Alice selects a random number c_1 to compute

$$Z' = E[\max\{a, b\} - c_1] = \frac{Z}{E[c_1]}.$$

Bob receives Z' from Alice. They jointly decrypt Z'. Bob then learns $c_2 = \max\{a, b\} - c_1$. The statistical indistinguishability of c_1 and that of c_2 can be proven by using a similar proof as in Lemma 4.1.

4.1.8.1 The Protocol Analysis

We now analyze the protocol for its complexity and security.

Time Complexity. We measure the time complexity by modular exponentiations, since they consume most of the time. Steps 1 and 2 require 12 modular exponentiations. The initialization of CMP (Ishai et al., 2003; Naor and Pinkas, 2001) also needs some modular exponentiations. However, the initialization can be done before the protocol, and its cost can be amortized over all the runnings of secure bit-length. Thus, we do not count its cost in Step 3. Alice and Bob jointly decrypt M' that involves 10 modular exponentiations in the Step 3. Step 4 takes 12 modular exponentiations. Therefore, the time complexity of secure max is 34 bounded by $O(1)$.

Communication Complexity. We measure the communication complexity by the number of message bits passing between Alice and Bob. Steps 1 and 2 need to transfer $10t_2$ bits where t_2 is the message length in Paillier cryptosystem. Step 3 needs $4t_2 + 3\ell t_1$ bits where ℓ is the bit-length of the $\max(M', r)$ and t_1 is a security parameter in Paillier cryptosystem (Note: t_1 is suggested to be 80 in practice (Kolesnikov et al., 2009)). The CMP

initialization also has some communication cost. We do not involve it, since it can be done before running the protocol in Step 3. The last step needs to transfer $6t_2$ bits. The communication cost of secure max is $20t_2 + 3\ell t_1$ bits bounded by $O(t_2 + \ell t_1)$.

Secure max is proven secure via simulation paradigm (Section 2.6.1) in the following.

Theorem 4.6 *The secure max protocol is simulatable.*

Proof 4.15 *We simulate the view of Alice and that of Bob. We first simulate the view of Alice. Let a be the input of Alice, and c_1 be the protocol output to her. According to the secure max protocol in Figure 4.6, the view of Alice is $VIEW_1^{\max} = (a, \mathcal{V}_1^1, \mathcal{V}_1^2, \mathcal{V}_1^3, \mathcal{V}_1^4)$, where $\mathcal{V}_1^1$ is the set of messages Alice receives from Bob for the CMP comparison in Step 3, $\mathcal{V}_1^2$ is $D[M]$, that is decryption of M in Step 3, $\mathcal{V}_1^3$ is the set of messages she receives for (2,2)-threshold Paillier decryption in Step 4, and $\mathcal{V}_1^4$ is the set of all the other messages she receives. The simulator $S_1^{\max}(a, c_1)$ to simulate $VIEW_1^{\max}$ is created as follows. The simulation of $\mathcal{V}_1^1$ and that of $\mathcal{V}_1^3$ are already given in (Kolesnikov and Schneider, 2008) and (Cramer et al., 2001), respectively. Thus, $S_1^{\max}$ can call simulators in (Kolesnikov and Schneider, 2008) and (Cramer et al., 2001) to simulate $\mathcal{V}_1^1$ and $\mathcal{V}_1^3$, respectively. To simulate $D[M]$ in $\mathcal{V}_1^2$, the simulator can use a similar proof as in Lemma 4.3. Each message $E[m] \in \mathcal{V}_1^3$ is a (2,2)-threshold Paillier encryption. To simulate it, $S_1^{\max}$ computes an encryption $E[r]$, where r is a random value. Since (2,2)-threshold Paillier encryption is semantically secure, $E[m]$ and $E[r]$ are computationally indistinguishable.*

The view of Bob is simulated in the following. Let b be the input of Bob, and c_2 be the protocol output to him. According to the secure max protocol in Figure 4.6, the view of Bob is $VIEW_2^{\max} = (b, \mathcal{V}_2^1, \mathcal{V}_2^2, \mathcal{V}_2^3)$, where $\mathcal{V}_2^1$ is the set of messages Bob receives from Alice for the CMP comparison in Step 3, $\mathcal{V}_2^2$ is the set of messages he receives for (2,2)-threshold Paillier decryption in Step 4, and $\mathcal{V}_2^3$ is the set of all the other messages he receives. The simulator $S_2^{\max}(b, c_2)$ to simulate $VIEW_2^{\max}$ is created as follows. The simulation of $\mathcal{V}_2^1$ and that of $\mathcal{V}_2^2$ are already given in (Kolesnikov and Schneider, 2008) and (Cramer et al., 2001), respectively. Thus, $S_2^{\max}$ can call simulators in (Kolesnikov and Schneider, 2008) and (Cramer et al., 2001) to simulate $\mathcal{V}_2^1$ and $\mathcal{V}_2^2$, respectively. Each message $E[m_b] \in \mathcal{V}_2^3$ is a (2,2)-threshold Paillier encryption. To simulate it, $S_2^{\max}$ computes an encryption $E[r_b]$, where r_b is a random value. Since (2,2)-threshold Paillier encryption is semantically secure, $E[m_b]$ and $E[r_b]$ are computationally indistinguishable.

4.1.9 Secure Max Location

The secure max location operator is an SMC protocol. Let $M_1, M_2, \cdots, M_l$ be a list of values, where $M_i = M_i^A + M_i^B$ (for $i = 1, 2, \ldots, l$), and M_i^A and M_i^B are held by Alice and Bob respectively. The secure max location protocol is to find an index θ, such that M_θ (correspondingly $M_\theta^A + M_\theta^B$) is the maximum value in the list, while not disclosing any additional information. For example, in Naïve Bayes, the secure max location protocol can find the location θ of the maximum a posteriori probability (MAP) estimation which it can then be mapped to a class label (i.e., response value) by either Alice or Bob. The secure max location protocol is depicted in Figure 4.7.

Alice and Bob first configure a (2,2)-threshold Paillier cryptosystem. Let (pk, sk) be the public and private key pair of the cryptosystem. Suppose that sk_A and sk_B are the secret shares of Alice and Bob respectively, such that the shares combined together can recover sk. Let $E[.]$ and $D[.]$ be the encryption and decryption functions corresponding to pk, and (sk_A, sk_B), respectively. Alice and Bob carry out the protocol step by step as follows.

Step 1. Alice first generates a list of encrypted pairs, $\forall_{i=1}^{l}(E[i], E[M_i^A])$. She sends the list to Bob.

Step 2. For each pair $(E[i], E[M_i^A])$, Bob re-encrypts $E[i]$, and computes an encryption of $M_i^A + M_i^B$ by

$$\begin{cases} I_i = E[i + 0] = E[i] \times E[0] \\ V_i = E[M_i^A + M_i^B] = E[M_i^A] \times E[M_i^B]. \end{cases} \tag{4.61}$$

Bob shuffles the list and sends it back to Alice.

Step 3. Alice re-encrypts each pair (I_i, V_i) by $I_i = I_i \times E[0]$ and $V_i = V_i \times E[0]$. She then also shuffles the list.

The shuffling and re-encryption above ensure that neither Alice nor Bob is able to correlate (I_i, V_i) with i correctly with a confidence significantly larger than $1/l$ for $1 \leq i \leq l$. This is guaranteed by the security feature of Paillier encryption. Still, without loss of generality and also for the simplicity of notations, we assume that the shuffled list is in order, that is, $(I_1, V_1), (I_2, V_2), \cdots, (I_l, V_l)$.

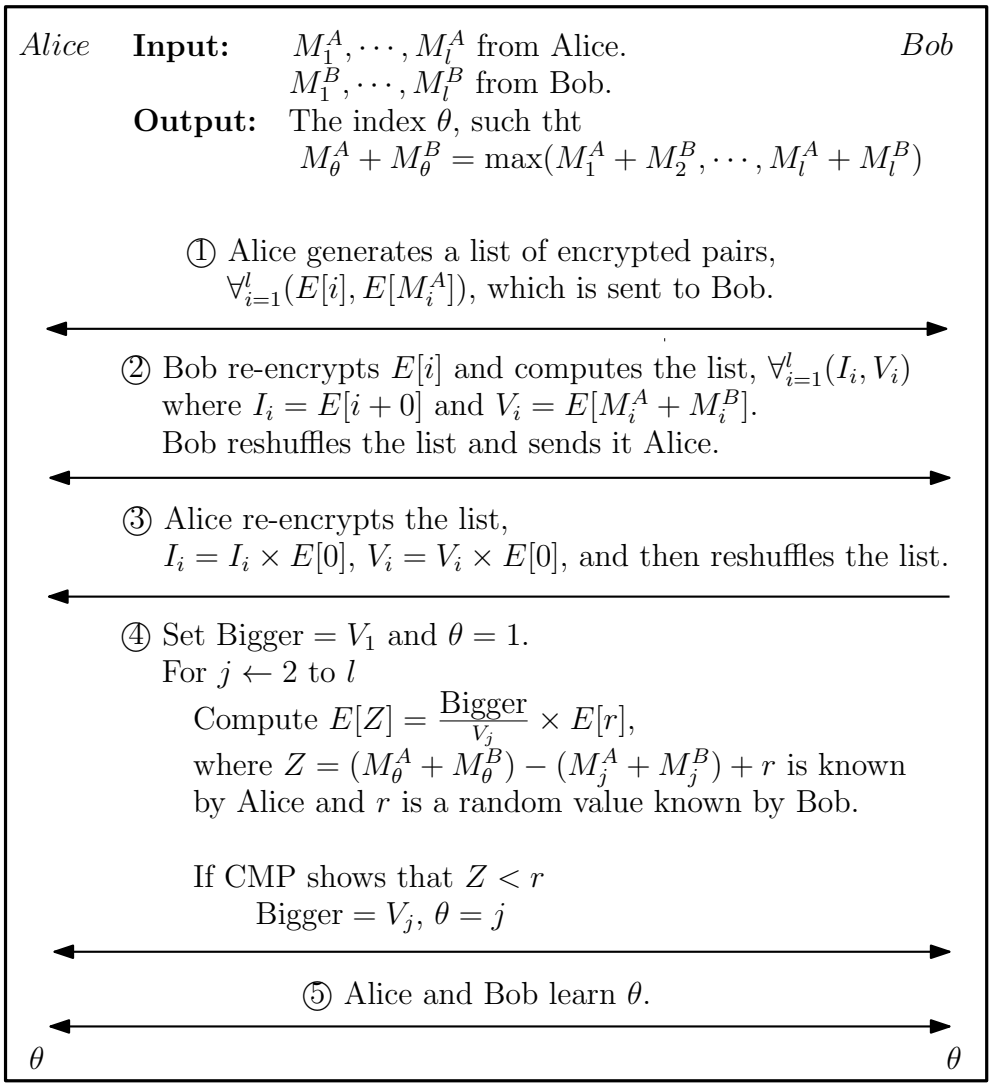


Figure 4.7: Secure max location protocol

Step 4. Alice and Bob locate the maximal value iteratively. They first compare $M_1^A + M_1^B$ with $M_2^A + M_2^B$. They take out the larger of the two, and compare it with $M_3^A + M_3^B$. The larger is then compared with $M_4^A + M_4^B$. This continues, until the comparison with $M_l^A + M_l^B$ is done. The details of this step are as follows. Initialize "Bigger" and θ to be V_1 and 1, respectively. For $j = 2, 3, \ldots, l$, Alice and Bob jointly compute $E[Z] = \frac{\text{Bigger}}{V_j} \times E[r_{j-1}]$, where $Z = (M_\theta^A + M_\theta^B) - (M_j^A + M_j^B) + r_{j-1}$ is known by Alice only and r_{j-1} is a random value known only by Bob. Alice and Bob then apply CMP (i.e., the secure integer comparison circuit in Section 3.1.10) to compare Z and r_{j-1}. If $Z < r_{j-1}$, then $(M_\theta^A + M_\theta^B) < (M_j^A + M_j^B)$, and the two parties set Bigger $= V_j$ and $\theta = j$.

Step 5. We assume that $(I_1, V_1), (I_2, V_2), \cdots, (I_l, V_l)$ are in order. Thus, the output θ from Step 4 gives the index of the biggest value. If the list is not in order, Alice and Bob can decrypt I_θ, and recover the index of the biggest value.

4.1.9.1 The Protocol Analysis

We now analyze the protocol for its complexity and security.

Time Complexity. We measure the time complexity by modular exponentiations, since they consume most of the time. Steps 1-3 requires $12l$ modular exponentiations. Step 4 needs $12(l-1)$ modular exponentiations. The initialization of CMP (Ishai et al., 2003; Naor and Pinkas, 2001) needs some modular exponentiations. However, the initialization can be run before the protocol, and its cost can be amortized over all the runnings of secure max. Thus, we do not count its cost in Step 4. The last step needs 10 modular exponentiation. Therefore, the number of modular exponentiations needed by secure max location is $24l - 2$ bounded by $O(l)$.

Communication Complexity. The communication cost of Steps 1-3 is $8t_2l$ bits, where t_2 is the message length in Paillier cryptosystem. Step 4 needs $3(\phi)t_1(l-1) + 8t_2(l-1)$ bits where ϕ is the number of bits of $\max(M_1^A + M_2^B, \cdots, M_l^A + M_l^B)$ and t_1 is a security parameter in Paillier cryptosystem (Note: t_1 is suggested to be 80 in practice (Kolesnikov et al., 2009)). The CMP initialization also has some communication cost. We do not involve it, since it can be done before running the protocol in Step 4. Step 5 transfers $4t_2$ bits. Therefore, the communication complexity is $16t_2l - 4t_2 + 3(\phi)t_1(l-1)$ bits bounded by $O(l(t_2 + t_1))$.

The secure max location protocol is proven secure via simulation paradigm (Section 2.6.1) in the following.

Theorem 4.7 *The secure max location protocol is simulatable.*

Proof 4.16 *We simulate the view of Alice and that of Bob. We first simulate the view of Alice. Let $M_1^A, \cdots, M_l^A$ be the inputs of Alice, and θ be the protocol output to her. According to the secure max location protocol in Figure 4.7, the view of Alice is $VIEW_1^{maxloc} = ((M_1^A, \cdots, M_l^A), \mathcal{V}_1^1, \mathcal{V}_1^2, \mathcal{V}_1^3, \mathcal{V}_1^4)$, where $\mathcal{V}_1^1$ is the set of messages Alice receives from Bob for the CMP comparison in Step 4, $\mathcal{V}_1^2$ is the set of the decrypted messages $\forall_{j=2}^{l} D[E[Z]]$*

where Alice receives from Bob in Step 4, , $\mathcal{V}_1^3$ is the set of messages she receives for (2,2)-threshold Paillier, decryption in Step 5, and $\mathcal{V}_1^4$ is the set of all the other messages she receives. The simulator $S_1^{maxloc}((M_1^A, \cdots, M_l^A), \theta)$ to simulate $VIEW_1^{maxloc}$ is created as follows. The simulation of $\mathcal{V}_1^1$ and that of $\mathcal{V}_1^3$ are already given in (Kolesnikov and Schneider, 2008) and (Cramer et al., 2001), respectively. Thus, S_1^{maxloc} can call simulators in (Kolesnikov and Schneider, 2008) and (Cramer et al., 2001) to simulate $\mathcal{V}_1^1$ and $\mathcal{V}_1^3$, respectively. To simulate $\mathcal{V}_1^2$, S_1^{maxloc} can use a similar proof as in Lemma 4.3. Each message $E[m] \in \mathcal{V}_1^4$ is a (2,2)-threshold Paillier encryption. To simulate it, S_1^{maxloc} computes an encryption $E[r]$, where r is a random value. Since (2,2)-threshold Paillier encryption is semantically secure, $E[m]$ and $E[r]$ are computationally indistinguishable.

Now we simulate the view of Bob. Let $M_1^B, \cdots, M_l^B$ be the inputs of Bob, and θ be the protocol output to him. The view of Bob is $VIEW_2^{maxloc} = (b, \mathcal{V}_2^1, \mathcal{V}_2^2, \mathcal{V}_2^3)$, where $\mathcal{V}_2^1$ is the set of (2,2)-threshold Paillier encrypted messages Bob receives from Alice in Step 2, $\mathcal{V}_2^2$ is the set of messages Bob receives from Alice for the CMP (Fast Garbled Circuit) in Step 4, and $\mathcal{V}_2^3$ is the set of messages Bob receives for (2,2)-threshold Paillier decryption in Step 4 and 5. The simulator $S_2^{maxloc}((M_1^B, \cdots, M_l^B), \theta)$ to simulate $VIEW_2^{maxloc}$ is created as follows. Each message $E[m_b] \in \mathcal{V}_2^1$ is a (2,2)-threshold Paillier encryption. To simulate it, S_2^{maxloc} computes an encryption $E[r_b]$, where r_b is a random value. Since (2,2)-threshold Paillier encryption is semantically secure, $E[m_b]$ and $E[r_b]$ are computationally indistinguishable. The simulations of $\mathcal{V}_2^2$ and $\mathcal{V}_2^3$ are already given in (Kolesnikov and Schneider, 2008) and (Cramer et al., 2001), respectively. Thus, S_2^{maxloc} can call the simulators in (Kolesnikov and Schneider, 2008) and (Cramer et al., 2001) to simulate $\mathcal{V}_2^2$ and $\mathcal{V}_2^3$, respectively.

4.1.9.2 The Protocol Extension

The protocol can be extended to find the maximum value of the list by only modifying the Step 5 in Figure 4.7. Alice selects a random number c_1 to compute

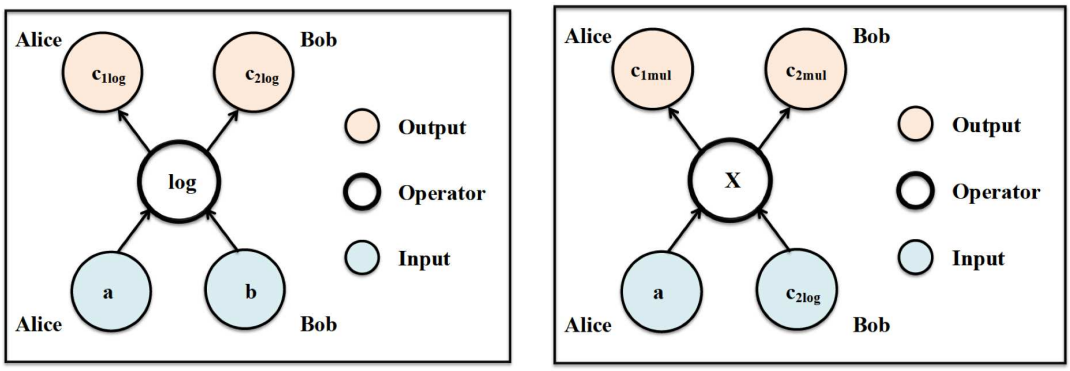
$$V = E[M_\theta^A + M_\theta^B - c_1] = \frac{Bigger}{E[c_1]}, \tag{4.62}$$

which is sent to Bob. Alice and Bob then jointly decrypt $D[V]$ where $V = M_\theta^A + M_\theta^B - c_1$. At the end of the protocol, Alice learns only c_1 and Bob learns only $c_2 = M_\theta^A + M_\theta^B - c_1$

where $c_1 + c_2 = M_\theta^A + M_\theta^B$ is the maximum value of the list. The time complexity and the communication complexity of the protocol extension are $24l$ (bounded by $O(l)$) and $16t_2l - 2t_2 + 3(\phi)t_1(l-1)$ (bounded by $O(l(t_2+t_1))$), respectively. Since the security proof of the protocol extension is similar in secure max location (Section 4.1.9), we omit the details here.

4.2 The DAG Model Analysis

In section, we first discuss the connection of secure operators of the DAG model. Finally, we analyze the security of our DAG model. Secure operators can be connected to serve for different functions. The connection of our secure operators of DAG can work in such a similar way of the connection of the arithmetic operators. The outputs of secure operators are different from the arithmetic operation; e.g. the connection of secure operations of DAG produces two outputs at the end of the execution. We use two examples to illustrate the workings of the connection of secure operators of DAG.



(a) Secure log

(b) Secure multiplication

Figure 4.8: Secure computations on $a \times \log(a+b)$.

Multiplication with log. Let a and b be inputs of Alice and Bob respectively. Alice and Bob jointly compute $a \times \log(a+b)$ as follows. They first apply secure log to compute $\log(a+b)$ to get c_{1log} and c_{2log}, held by Alice and Bob respectively, as depicted in Figure 4.8(a). Alice and Bob then apply secure multiplication to compute $a \times c_{2log}$. The outputs of the secure multiplication are c_{1mul} and c_{2mul}, held by Alice and Bob respectively, as

depicted in Figure 4.8(b). Thus,

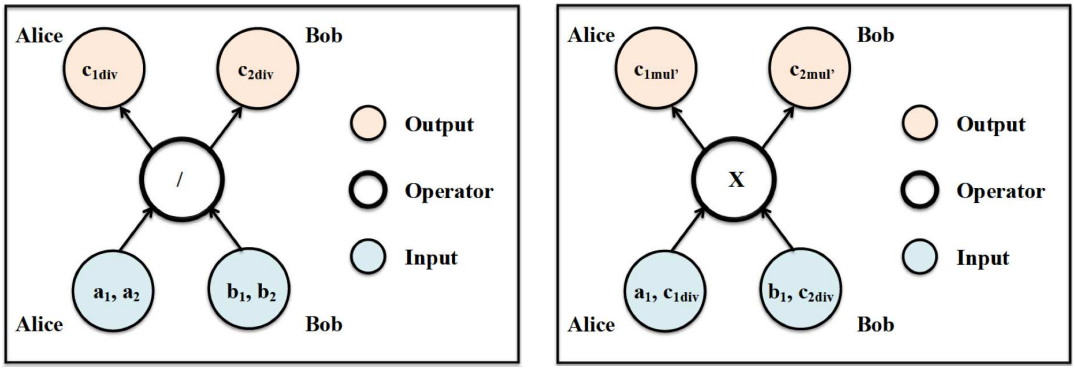
$$a \times \log(a+b) = a \times (c_{1log} + c_{2log}) = ac_{1log} + (a \times c_{2log})$$
$$= (ac_{1log} + ac_{1mul}) + ac_{2mul} = c_1 + c_2, \tag{4.63}$$

where $c_1 = ac_{1log} + ac_{1mul}$ and $c_2 = ac_{2mul}$, held by Alice and Bob respectively, are the outputs of the computation $a \times \log(a+b)$. Since Alice holds a and c_{1log}, she can multiply a by c_{1log} without involving any secure operator.

Multiplication with division. Let (a_1, a_2) and (b_1, b_2) be inputs of Alice and Bob respectively. Alice and Bob jointly compute $\frac{a_1+b_1}{a_2+b_2} \times (a_1+b_1)$ as follows. They first apply secure division to compute $\frac{a_1+b_1}{a_2+b_2}$ to get c_{1div} and c_{2div}, held by Alice and Bob respectively, as depicted in Figure 4.9(a). Alice and Bob then apply secure multiplication to compute two vectors, $[c_{1div}\ a_1]^T$ and $[b_1\ c_{2div}]^T$. The outputs of the secure multiplication are $c_{1mul'}$ and $c_{2mul'}$, held by Alice and Bob respectively, as depicted in Figure 4.9(b). Thus,

$$\frac{a_1+b_1}{a_2+b_2} \times (a_1+b_1) = (c_{1div} + c_{2div}) \times (a_1+b_1) = a_1c_{1div} + \begin{bmatrix} c_{1div} \\ a_1 \end{bmatrix} \times \begin{bmatrix} b_1 \\ c_{2div} \end{bmatrix} + b_1c_{2div}$$
$$= (a_1c_{1div} + c_{1mul'}) + (c_{2mul'} + b_1c_{2div}) = c'_1 + c'_2, \tag{4.64}$$

where $c'_1 = a_1c_{1div} + c_{1mul'}$ and $c'_2 = c_{2mul'} + b_1c_{2div}$, held by Alice and Bob respectively, are the outputs of the computation $\frac{a_1+b_1}{a_2+b_2} \times (a_1+b_1)$. Since Alice holds a_1 and c_{1div}, she can multiply a_1 by c_{1div} without involving any secure operator. Likewise Bob can multiply b_1 by c_{2div} which both of them are known by him.



(a) Secure division (b) Secure multiplication

Figure 4.9: Secure computations on $\frac{a_1+b_1}{a_2+b_2} \times (a_1+b_1)$.

Therefore, secure operators of DAG model can be connected to serve various functions. Their connection is similar to the connection of arithmetic operators.

Analysis of the Connection of Secure Operators. We now analyze the error when operators are connected. Without loss of generality, we consider the connection of two operators. We first consider that an operator is connected to secure division, that is, its output is input to secure division. Without loss of generality, suppose that the operator is multiplication. Let (a_1, a_2) be the input of Alice and (b_1, b_2) be that of Bob. Suppose that the connection of multiplication and division is $\frac{a_1 \times b_1}{a_2 + b_2} = (a_1 \times b_1)\frac{1}{a_2+b_2}$. Then, $a_1 \times b_1$ is approximated to $(a_1 \times b_1)(1 - e_t)$, and $\frac{1}{a_2+b_2}$ is approximated to $(a_2 + b_2)(1 - e_d)$, where e_t is the error for secure multiplication (Lemma 4.2) and e_d is the error of secure division (Lemmas 4.4 and 4.5). Thus, the relative error of $\frac{a_1 \times b_1}{a_2+b_2}$ is

$$1 - (1 - e_t)(1 - e_d) \approx e_t + e_d, \tag{4.65}$$

where $e_t \cdot e_d$ is truncated since it is much smaller than e_t and e_d.

The error analysis of the connection of multiplication and division can be easily extended to that of log/power and division. Let e_l be the error of secure log, and e_p be the error of secure power. If log (or power) is connected to division, then the error of the two operators can be approximated to $|-e_l + e_d|$ (or $|\pm e_p + e_d|$).

Next, we study the error of an operator connected to log. Again, we again assume that the operator is multiplication. Let a and b be the inputs of Alice and Bob respectively, and the connection of the two operators is $\log(a \times b)$. Then, $a \times b$ is approximated to $(a \times b)(1 - e_t)$, where e_t is the error for secure multiplication (Lemma 4.2). Let e_l be the approximation error of secure log (Lemmas 4.6 and 4.7). Then, the relative error of $\log(a \times b)$ is

$$\begin{aligned}
&\left|1 - \frac{\log\left((a \times b)(1 - e_t)\right)}{\log(a \times b)}(1 + e_l)\right| \\
=&\left|1 - \frac{\log(a \times b) + \log(1 - e_t)}{\log(a \times b)}(1 + e_l)\right| \\
\approx&\left|1 - \frac{\log(a \times b) - e_t}{\log(a \times b)}(1 + e_l)\right| \\
\approx&\left|-e_l + \frac{e_t}{\log(a \times b)}\right|.
\end{aligned} \tag{4.66}$$

In the above, the first approximation is obtained since $\log(1-e_t) \approx -e_t$ for small e_t, and the second approximation is obtained since $e_t \cdot e_l$ is much smaller than e_t and e_l. Similar to the extension of connecting other operators with division, we can connect another operator other than multiplication with log. The error can be approximated to the linear combination of the errors of the two connected operators as approximate Equation 4.66.

Finally, we investigate the connection between an operator and power. Once again, we take multiplication as the operator. The connection by another operator can be studied similarly. Let a and b be the inputs of Alice and Bob, respectively, and $(a \times b)^{\frac{m}{d}}$ be the connection of the two operators. $a \times b$ is approximated to $(a \times b)(1-e_t)$, where e_t is the error for secure multiplication (Lemma 4.2). Let e_p be the error of secure power (Lemmas 4.8 and 4.9). The relative error of $(a \times b)^{\frac{m}{d}}$ is

$$\begin{aligned} & \left| 1 - \frac{((a \times b)(1-e_t))^{\frac{m}{d}}}{(a \times b)^{\frac{m}{d}}}(1 \pm e_p) \right| \\ \approx & \left| 1 - \frac{(a \times b)^{\frac{m}{d}}(1-\frac{m}{d}e_t)}{(a \times b)^{\frac{m}{d}}}(1 \pm e_p) \right| \\ \approx & \left| \pm e_p + \frac{m}{d}e_t \right|, \end{aligned} \tag{4.67}$$

where the first approximation holds as $(1-e_t)^{\frac{m}{d}} \approx 1 - \frac{m}{d}e_t$ for small e_t, and the second approximation holds as $e_t \cdot e_p$ is much smaller than e_t and e_p.

The error analysis above shows that the connection of two secure operators leads to a relative error, which is approximately bounded by the linear combination of the relative errors of the two operators. Easily, we can extend the discussion to the connection of more operators, and the relative error can again be bounded by the linear combination of the errors of the operators.

The Security of Connected Operators. We analyze the security of our DAG model. We propose 9 secure operators: secure multiplication, secure bit-length, secure division, secure log, secure power, secure max, and secure max location are simulatable in Theorem 4.1, Theorem 4.2 Theorem 4.3, Theorem 4.4, Theorem 4.5, Theorem 4.6 and Theorem 4.7, respectively. Another two operators, secure addition and secure minus are trivial, which Alice and Bob do not even pass messages to each other (i.e., require no interaction). Thus, secure addition and secure minus can be seen as special SMC protocols, and they are simulatable. Thus, 9 secure operators of DAG model are strictly proven secure.

In DAG, we can pipeline multiple operators to perform more complex functions. Since each secure operator is simulatable as discussed previously, the connection of secure operators is also simulatable. In the connection of the multiple operators, the invocation of secure operators is sequential. The inputs of the operator but the first operator are the outputs of the previous operators. The connection of the multiple operators is simulatable via composition theorem (Lindell and Pinkas, 2008). The theorem below proves that the connection of the multiple operators will allow Alice and Bob to learn only the protocol output with their respective inputs.

Theorem 4.8 *In the* DAG *model, Alice (Bob) only learns the protocol output to her (him).*

Proof 4.17 *Let $op_1, op_2, \cdots, op_k$ be the nodes (secure operators) in a directed acyclic graph. Operator op_i (for $i = 1, 2, \ldots, k$) outputs c_1^i and c_2^i, held by Alice and Bob, respectively. The outputs of all the operators except for the last one are intermediate results, and the output of the last one (say op_k) is the model output.*

We first simulate the view of Alice. Let A be the input of Alice and c_1^k (i.e., one output of op_k) be the protocol output to her. The view of Alice is $VIEW_1^{\mathsf{DAG}} = (A, \mathcal{V}_1^1, \mathcal{V}_1^2, \cdots, \mathcal{V}_1^k)$, where $\mathcal{V}_1^i$ is the set of messages she receives from Bob when running operator op_i for $1 \leq i \leq k$. If op_i is addition or minus, then $\mathcal{V}_1^i$ is empty and c_1^i (generated by Alice alone) is not included in the view. The simulator $S_1^{\mathsf{DAG}}(A, c_1^k)$ to simulate $VIEW_1^{\mathsf{DAG}}$ is created as follows. The simulation of $\mathcal{V}_1^i$ is done when discussing each secure operator. Thus, S_1^{DAG} can call a corresponding simulator to simulate it.

Next, we simulate the view of Bob. Let B be the input of Bob and c_2^k (i.e., another output of op_k) be the protocol output to him. The view of Bob is $VIEW_2^{\mathsf{DAG}} = (B, \mathcal{V}_2^1, c_2^1, \mathcal{V}_2^2, c_2^2, \cdots, \mathcal{V}_2^k)$, where $\mathcal{V}_2^i$ is the set of messages he receives from Alice when running operator op_i for $1 \leq i \leq k$. If op_i is addition or minus, then $\mathcal{V}_2^i$ is empty and c_2^i (generated by Bob alone) is not included in the view. The simulator $S_2^{\mathsf{DAG}}(B, c_2^k)$ to simulate $VIEW_2^{\mathsf{DAG}}$ is created as follows. It first call the simulators of single operators to simulate $\mathcal{V}_2^i$. To simulate immediate result c_2^i (for $i = 1, 2, \ldots, k-1$), S_2^{DAG} can select a random number r as in Lemma 4.1, and then r is statistically indistinguishable from c_2^i.

4.3 Experiment and Discussion

In this section, we evaluate the performance of our DAG model. We evaluate the secure operators, which consist of the model, with respect to their efficient and approximation errors.

Table 4.1: Experiment Parameters to Evaluate Secure Operators

Operator	κ	γ	β	τ	w	z	λ
$\times$	-	15	284	300	-	-	-
$/, \log$	950		12	28	20	-	40
power	910						
bit-length	-	-	-	-	-	45	-

Implementation. We implemented the solutions by Java. All experiments were run on a machine of Intel® Xeon® 2.3GHz CPU (16 cores) with 128.0GB RAM running on Windows Server 2012. We downloaded the threshold Paillier Cryptosystem [1], and configured it to 1024-bit. The CMP which is part of the fast garbled circuit (Huang, Evans, Katz and Malka, 2011) is one of the most efficient secure comparisons. The security parameters like κ are restricted by different conditions in different secure operators. We set them based on these conditions, and thus their values vary from one operator to another. Table 4.1 gives the configuration details.

Evaluation. We first evaluate secure bit-length that is the sub-protocol of secure division, secure log and secure power. The operator can run in parallel. We thus implement it in two versions – the first runs the whole protocol (in Figure 4.2) sequentially, and the second executes the protocol in parallel using 20 threads (Step 3 in Figure 4.2). Figure 4.10(a) reports the results when varying the number of times of running the secure bit-length operator. Clearly, the parallel version with 20 threads is more efficient. On average the elapsed time per operation is 38.5820 sec by the sequential version, while that by the parallel version is only 4.2385 sec.

On the basis of the bit-length protocol, we then study the elapsed time of secure division, secure log, and secure power. Figure 4.10(b) is the results when the operators apply sequential version of the underlying bit-length protocol, and Figure 4.10(c) is the results of parallel version of the bit-length protocol. As expected, the efficiency of parallel

[1] http://www.utdallas.edu/∼mxk093120/paillier/

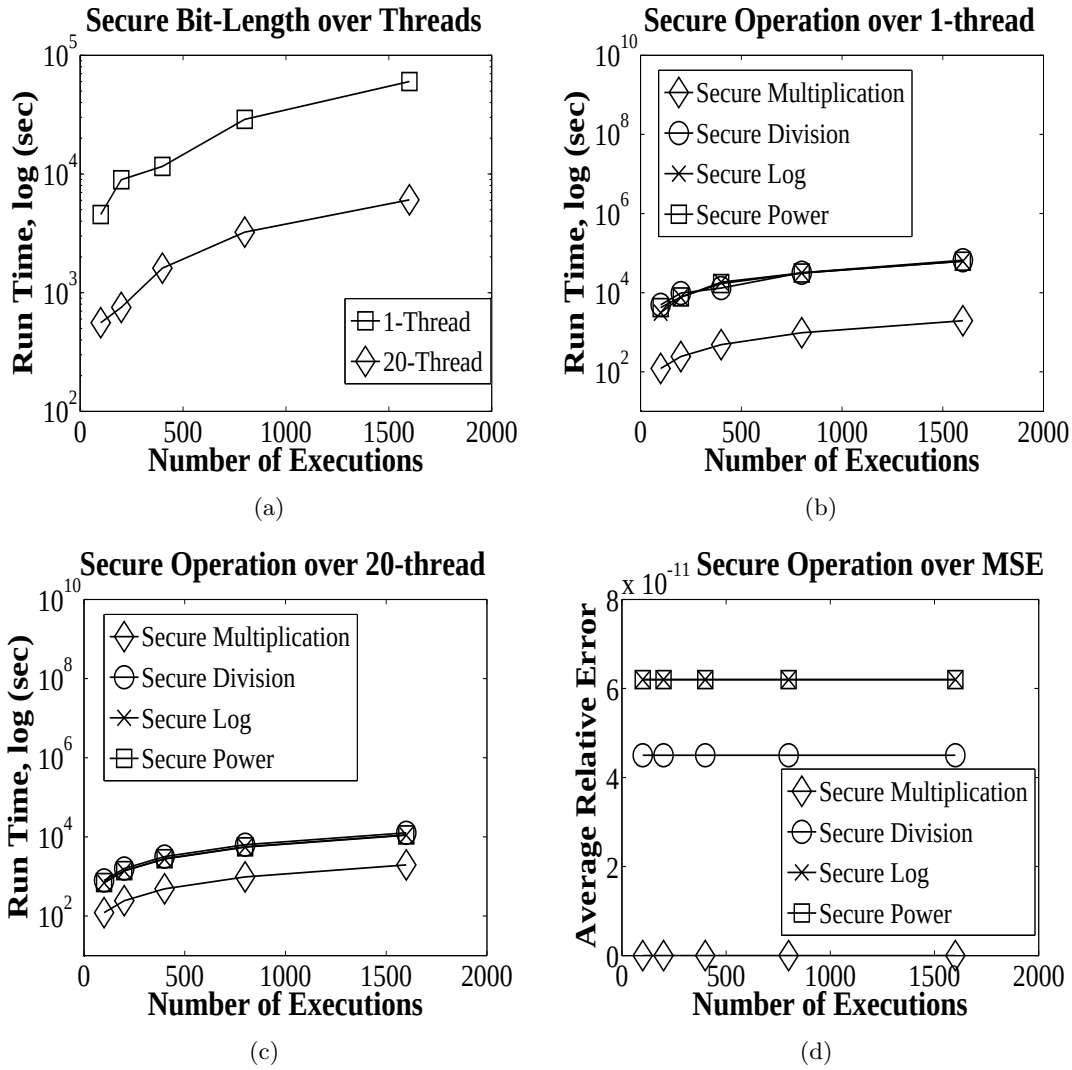


Figure 4.10: Performance of the Secure Operators

running is much better. For example, one execution of secure division based on sequential secure bit-length protocol takes 42.4461 sec, and that on parallel version takes only 7.9312 sec. In the above two figures, we have also included the experimental results of secure multiplication. The protocol is simpler, and takes around 1.218 sec. Since it does not depend on bit-length sub-protocol, its running time does not change obviously whether the bit-length sub-protocol is running either in sequential or in parallel.

The errors are small, regardless of the number of execution threads – on average the error of secure multiplication is 7.3266×10^{-92}, that of secure division is 4.4981×10^{-11}, that of secure log is 6.2039×10^{-11}, and that of secure power is 7.7770×10^{-11}. This is consistent with the setting of security parameters (i.e., β and w) and the developed error bounds (i.e., secure multiplication in Lemma 4.2, secure division in Lemmas 4.4 and 4.5, secure log in Lemmas 4.6 and 4.7, and secure power in Lemmas 4.8 and 4.9).

4.4 Chapter Summary

In this chapter, we propose DAG – a general model for privacy preserving data mining. Currently our DAG model consists of 9 secure operators: secure addition, secure minus, secure multiplication, secure division, secure log, secure power, secure bit-length, secure max and secure max location. The secure operators of DAG model can be pipelined together to compute a complex function. The protocol analysis of each operator is discussed in detail such as its approximation errors if applicable, complexity and security. The error bound of the concatenation of operators is derived based on the error bound of every single operator. Secure operators of DAG are strictly proven secure via simulation paradigm (Goldreich, 2004). Thus, our DAG can provide a complete privacy under the semi-honest model. The performance of secure operators are evaluated in the experiment. The theoretical and experimental proofs show that our DAG model is efficient in computation and effective in protecting data privacy.

Chapter 5

Privacy-Preserving Classification Algorithms by DAG

Data mining consists of three key tasks: regression, information retrieval, and clustering on data to yield a data driven model. Specifically, the tasks help to discover patterns in data by analyzing large quantities of data. In data mining, clustering or classification predicts a certain outcome based on a given input. Classification algorithms process training datasets that contain a set of attributes (predictors) with their class attribute (response variable) to predict the outcome. The algorithms then process testing datasets that contain the same set of predictors to predict response value.

Data are distributed across multiple parties in many data mining applications. However, data may contain sensitive information; directly sharing it could violate personal privacy (Aggarwal and Yu, 2008; Clifton et al., 2002) and privacy laws (Congress, 1996; ECHR, 2014). Therefore, data sharing for data mining tasks needs to be carried out in a privacy-preserving way. Secure multi-party computation (SMC) protocols have been extensively applied in privacy-preserving data mining (Lindell and Pinkas, 2008; Verykios, Bertino, Fovino, Provenza, Saygin and Theodoridis, 2004) in the context of multiple parties. It is assumed that these parties are not willing to share their data directly, but would like to work together to learn the output of any agreed mining task. Various SMC techniques have been proposed to serve different mining tasks, such as decision tree (Vaidya, Clifton, Kantarcioglu and Patterson, 2008), Naïve Bayes (Vaidya, Kantarcioglu and Clifton, 2008), support vector machine (Teo et al., 2013) and singular value decomposition (Han et al., 2009). However, most of these techniques (Bertino et al., 2005; Aggarwal and Yu, 2008) only consider secure addition and secure multiplication operations; they cannot be applied to the cases that include more complicated operations such as secure division. In addition, these techniques are ad-hoc, and thus difficult to be directly applied to mining tasks other than those they target.

Thus, we propose the DAG model (Chapter 4) that can be applied in various data mining tasks. In this chapter, we apply the DAG model into three different classification algorithms: support vector machine (Burges, 1998) kernel regression (Friedman et al., 2001) and Naïve Bayes (Mitchell, 1997). We assume two parties, Alice and Bob that are semi-honest, in each classification task. Specifically, in kernel regression and Naïve Bayes, each party has a private subset of data (i.e., training data) with the format $(x_1, x_2, \ldots, x_d, y)$, where x_i's are predictor variables and y is the response variable. We also assume that Alice and Bob have another set of data (i.e., testing data) with only the x_i's values, and they would like to predict the y values by either kernel regression or Naïve Bayes. To protect the data privacy of the two parties, we build the data mining tasks based on the model. Extensive experimental results show that the response values predicted by DAG are very close to those predicted in non-private setting, where Alice and Bob simply disclose their data. In the kernel regression, the difference by MSE on predicted values between DAG and non-private setting is less than 0.0002. For Naïve Bayes, the outputs from the two cases are identical. The experimental results also show that the running time of our DAG model is efficient. For example, in kernel regression, when training data size of Alice (Bob) is 683,093, one prediction in non-private setting takes 5.93 sec, and that by our DAG model takes 12.38 sec.

In the following, we first discuss support vector machine integrated with DAG in Section 5.1. Kernel regression and Naïve Bayes integrated with DAG are discussed in Sections 5.2 and 5.3, respectively. Lastly, we summarize this chapter in Section 5.4.

5.1 Support Vector Machine (SVM)

A basic classification of linear support vector machine (SVM) can find a linear boundary by taking a set of input data and predict two possible responses (classes) from the inputs.

$$y = w.x + b. \tag{5.1}$$

The linear SVM can find a separating hyperplane that maximizes a margin by a straight line as in Equation 5.1. The margin is the distance between the hyperplane and the closest data points (i.e., support vectors). In the non-linear (kernel) classification, SVM uses some kernel functions to separate the input data into two possible responses by

$$P = \begin{pmatrix} d_1 & d_2 & d_3 & d_4 \\ d_5 & d_6 & d_7 & d_8 \\ d_9 & d_{10} & d_{11} & d_{12} \\ d_{13} & d_{14} & d_{15} & d_{16} \end{pmatrix} \tag{5.2}$$

Figure 5.1: Data in the 4×4 square matrix.

mapping input data into high dimensional feature spaces. Some popular kernel functions such as the polynomial kernel function, $K(x, y) = (1 + x.y)^s$ and the RBF kernel function, $K(x, y) = \exp(\frac{-1(x-y)^2}{2\sigma^2})$.

Therefore, kernel matrix is the main structure that contains all required kernel information for a learning algorithm in the non-linear SVM. In other words, the kernel matrix can play an intermediate role to generate a global SVM model without disclosing any local data. We assume that data are partitioned arbitrary between Alice and Bob: data can be randomly split on either horizontally or vertically, and even both of them. To securely construct the global SVM model without disclosing any data of Alice and that of Bob, we apply our DAG model to securely compute the Gram (i.e., kernel) matrix which is the dot product of the data vectors of Alice and that of Bob. The algorithm of SVM to compute the Gram matrix is briefly discussed in Section 5.1.1. We propose privacy-preserving SVM (PPSVM) by our DAG model and discuss the security analysis and the complexity analysis of PPSVM in Sections 5.1.2 and 5.1.3, respectively. Lastly, we evaluate the performance of PPSVM in Section 5.1.4.

5.1.1 Algorithm

In SVM, we can present data in a square matrix form. For example, in Figure 5.1, matrix P contains four tuples and four attributes, indicated by row and by column respectively. Each tuple of P has four data points. The kernel (Gram) matrix computes the dot products of every data pair in P: e.g., $[d_1\ d_2\ d_3\ d_4] \cdot [d_1\ d_2\ d_3\ d_4]$, $[d_1\ d_2\ d_3\ d_4] \cdot [d_5\ d_6\ d_7\ d_8]$, $[d_1\ d_2\ d_3\ d_4] \cdot [d_9\ d_{10}\ d_{11}\ d_{12}]$, and $[d_1\ d_2\ d_3\ d_4] \cdot [d_{13}\ d_{14}\ d_{15}\ d_{16}]$. Thus, the gram matrix of P is equivalent to $P \cdot P^T$. Any non-squared data matrix are converted into a square matrix by inserting zero into the matrix; e.g, in the 3×4 data matrix, we insert a row of [0 0 0 0] to make it the 4×4 data square matrix.

We discuss a few methods that can calculate the gram matrix in the following. We first propose a method, the upper/lower matrix multiplication based on the observation of

the Gram matrix multiplication. Some improved matrix multiplication methods such as Strassen multiplication (Strassen, 1969), Strassen-Coppersmith-Winograd multiplication (Coppersmith and Winograd, 1987), and Strassen-Bodrato multiplication (Bodrato, 2010) can also compute the Gram matrix.

- **the upper/lower matrix multiplication:** Given a matrix A, the Gram matrix $\mathcal{G}$ is the dot product of A with its transpose A^T as follows,

$$\mathcal{G}_{ij} = (AA^T)_{ij} = \sum_{k=1}^{n} a_{ik}(a^T)_{kj}, \tag{5.3}$$

  where i and j are the position of row (i.e., tuple) and that of column (i.e., attribute) , respectively, and n is the number of tuples. From Equation 5.3, we observe that $(AA^T)_{ij}$ shows the symmetric property which $(AA^T)_{ij}$ is equivalent to $(AA^T)_{ji}$. Thus, the upper/lower matrix multiplication can reduce at least 1/3 operations of n^3 multiplications and that of $n^2(n+1)$ linear operations (additions/subtractions).

- **Strassen multiplication:** Strassen's method (Strassen, 1969) can compute dot product of square matrices. The method increases addition operations as compared to the normal dot product computation. However, the method can reduce the number of expensive multiplication operations. For example, in the 2×2 data square matrix P, the normal dot product of P and P^T requires 4 additions and 8 multiplications. In contrast, Strassen's method needs 18 additions and 7 multiplications. The complexity of Strassen's method is $O(n^{2.807})$ (Strassen, 1969).

- **Strassen-Coppersmith-Winogard multiplication:** (Coppersmith and Winograd, 1987) improve the Strassen multiplication by reducing the number of addition operations. Again, in the 2×2 data square matrix P, the dot product of P and P^T requires 15 additions in (Coppersmith and Winograd, 1987) as compared to 18 additions in the Strassen's method. Both of the two methods have the same number of multiplications (i.e., 7 multiplications). Thus, the complexity of Strassen-Coppersmith-Winogard multiplication is $O(2^{2.374})$.

- **Strassen-Bodrato multiplication:** (Bodrato, 2010) proposes a Strassen-like method that can compute dot product of matrices. To reduce computation complexity in the multiplication, (Bodrato, 2010) introduces a new sequence technique that uses

Algorithm 1: Privacy-Preserving Support Vector Machine (PPSVM) on arbitrarily partitioned data

Input: Matrix $H = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix}$ has $n \cdot n$ data points that are split arbitrarily between Alice and Bob. Let H_a be some data points of H for Alice and H_b be some data points of H for Bob where $H_a + H_b = H$.

Output: Alice gets c_1 and Bob gets c_2 where $c_1 + c_2 = H \cdot H^T = \mathcal{G}$ (Gram matrix).

1 Alice generates a pair of keys (sk, pk) (i.e., (secret key, public key)).
2 Alice sends pk to Bob.
3 Alice and Bob generate zero matrices $(n \times n)$, O_a and O_b, held by Alice and Bob respectively.
4 Alice adds O_a into H_a to get $H'_a = H_a + O_a$.
5 **for** *i = 1 to n* **do**
6 **for** *j = 1 to n* **do**
7 Alice encrypts $E[d^a_{ij}]$ of H'_a which is sent to Bob.
8 **end**
9 **end**
10 Bob adds O_b into H_b to get $H'_b = H_b + O_b$, and generates a random matrix R.
11 **for** *i = 1 to n* **do**
12 **for** *j = 1 to n* **do**
13 Bob encrypts $Q_{ij} = E[d^a_{ij}] \times E[d^b_{ij} + r_{ij}]$ which is sent to Bob.
14 **end**
15 **end**
16 Bob also generates the second random matrix R' to compute $H_r = \frac{E[R^2]}{(Q^T)^R \times Q^R \times E[R']}$, which is sent to Alice. Bob sets $c_2 = R'$.
17 Alice decrypts $D[Q]$ and $D[H_r]$ to compute $c_1 = D[Q] \cdot (D[Q])^T + D[H_r]$ where $c_1 + c_2 = H \cdot H^T = \mathcal{G}$.

the squaring and complex dot product. The method has the same computation complexity of the Strassen-Coppersmith-Winogard multiplication. Thus, the complexity of Strassen-Bodrato multiplication is $O(2^{2.374})$.

5.1.2 Privacy-Preserving Support Vector Machine (PPSVM)

We propose privacy-preserving Support Vector Machine (PPSVM) that can securely compute the Gram matrix $\mathcal{G}$ that involves dot product of square matrices. PPSVM can apply secure multiplication operator of DAG to compute $\mathcal{G}$ as depicted in Algorithm 1. We assume two parties, Alice and Bob, in our PPSVM, that are semi-honest but curious - they strictly follow the protocol and will not collude with each other.

Let H be a $n \times n$ data square matrix as follows,

$$H = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix}, \tag{5.4}$$

where $\forall_{i=1,j=1}^{n} d_{ij}$ are data points in the dataset, j represents the attribute of the dataset, and i represents the location of the tuple (i.e., the tuple consists of n data points). Thus, H consists of n tuples with n attributes. In most cases, the total number of attributes m is less than the total number of tuples n. To make a data square matrix, we set $\forall_{i=1,j=m+1}^{n} d_{ij} = 0$ in H. Let data points of H be split arbitrary between Alice and Bob. Alice configures Paillier cryptosystem to generate a pair of keys (pk, sk), which sk is the secret key and pk is the public key. Let $E[.]$ and $D[.]$ be the encryption and decryption functions corresponding to pk and sk, respectively. The public key pk is sent to Bob. They apply PPSVM to compute $\mathcal{G}$ of H step by step as follows.

Step 1. Let H_a be some data points of H for Alice and H_b be some data points of H for Bob. Alice and Bob generate zero matrices $(n \times n)$, O_a and O_b, held by Alice and Bob respectively. Alice adds O_a into H_a to get H'_a. Alice encrypts $E[H'_a]$ which is sent to Bob.

$$E[H'_a] = \begin{pmatrix} E[d^a_{11}] & E[d^a_{12}] & \cdots & E[d^a_{1n}] \\ E[d^a_{21}] & E[d^a_{22}] & \cdots & E[d^a_{2n}] \\ \vdots & \vdots & \vdots & \vdots \\ E[d^a_{n1}] & E[d^a_{n2}] & \cdots & E[d^a_{nn}] \end{pmatrix}. \tag{5.5}$$

Step 2. Bob generates a random matrix R and adds O_b into H_b to get H'_b. Bob then computes

$$Q = E[H'_a + H'_b + R] = E[H'_a] \times E[H'_b + R] = \forall_{i=1,j=1}^{n} E[d^a_{ij}] \times E[d^b_{ij} + r_{ij}], \tag{5.6}$$

where $r_{ij} \in R$. Bob sends Q to Alice. The dot product of $\frac{Q}{E[R]}$ and $(\frac{Q}{E[R]})^T$ is

$$\frac{Q}{E[R]} \cdot \left(\frac{Q}{E[R]}\right)^T = E[Q]^{Q^T} \times \frac{E[R^2]}{(Q^T)^R \times Q^R}, \tag{5.7}$$

where $Q = E[H'_a + H'_b + R]$. Bob selects a random matrix R' to compute $H_r = \frac{E[R^2]}{(Q^T)^R \times Q^R \times E[R']}$ which R and R' are known only by Bob. Bob can apply secure multiplication operator of DAG to compute H_r which is sent to Alice. Bob sets $c_2 = R'$.

Step 3. Alice decrypts $D[Q]$ and $D[H_r]$ to compute

$$c_1 = D[Q] \cdot (D[Q])^T + D[H_r], \tag{5.8}$$

where $D[Q] = H'_a + H'_b + R$ and $D[H_r] = R^2 - Q^T R - QR^T - R'$. At the end of the protocol, Alice learns only c_1 and Bob learns only c_2 where $c_1 + c_2 = H \cdot H^T$ is $\mathcal{G}$ (Gram matrix).

The number of cryptographic operations can be reduced based on the observation. For example, a tuple with m attributes in H where $m < n$, Alice and Bob only apply the cryptographic computation on m attributes instead of n in H. We next analyze the performance and the security of the PPSVM.

5.1.3 Security Analysis and Complexity Analysis

We use time complexity and communication complexity to measure the performance of the privacy-preserving Support Vector Machine (PPSVM).

Time Complexity. We measure the time complexity of PPSVM by modular exponentiations, since they consume most of the time. We assume n tuples with m attributes in H that is equal to $n \cdot m$ data points. In Step 1, Alice needs $2nm$ modular exponentiations to encrypt $E[H'_a]$. Bob needs $2nm$ modular exponentiations to compute Q and $8mn$ modular exponentiations to compute H_r. Thus, Step 2 needs $10mn$ modular exponentiations. Step 3 needs 2 modular exponentiations to decrypt $D[Q]$ and $D[H_r]$. Therefore, the number of modular exponentiations needed by PPSVM is $12nm + 2$ bounded by $O(nm)$.

Communication Complexity. We measure the communication complexity by the number of message bits passing between Alice and Bob. We assume n tuples with m attributes in H that is equal to $n \cdot m$ data points. In Step 1, Alice transfers $2nmt_2$ bits to Bob where t_2 is a security parameter value in Paillier encryption (e.g., t_2 =1024). Step 2 needs a total of $4nmt_2$ bits. Therefore, the communication complexity is $6nmt_2$ bits bounded by $O(nmt_2)$.

Our proposed PPSVM is proven secure via simulation paradigm (refer to Section 2.6.1 for more details) in the following.

Theorem 5.1 *The PPSVM protocol is simulatable.*

Proof 5.1 *We simulate the view of Alice and that of Bob. We first simulate the view of Alice. Let H_a be the input of Alice, and c_1 be the protocol output to her. The view of Alice is $VIEW_1^{\text{ppsvm}} = (H_a, \mathcal{V}_1^1, \mathcal{V}_1^2)$ where $\mathcal{V}_1^1$ is the set of messages she receives to compute H_r in Step 2 and $\mathcal{V}_1^2$ is the set of messages she receives for Paillier decryption in Step 3. The simulator $S_1^{\text{ppsvm}}(H_a, c_1)$ to simulate $VIEW_1^{\text{ppsvm}}$ is created as follows. The simulations of $\mathcal{V}_1^1$ and $\mathcal{V}_1^2$ are already given in Theorem 4.1 and (Cramer et al., 2001), respectively. Thus, S_1^{ppsvm} can call the simulators in Theorem 4.1 and (Cramer et al., 2001) to simulate $\mathcal{V}_1^1$ and $\mathcal{V}_1^2$ respectively.*

The view of Bob is simulated in the following. Let H_b be the input of Bob, and c_2 be the protocol output to him. The view of Bob is $VIEW_2^{\text{ppsvm}} = (H_b, \mathcal{V}_2^1, \mathcal{V}_2^2)$, where $\mathcal{V}_2^1$ is the set of encrypted messages he receives from Alice in Step 2 and $\mathcal{V}_2^2$ is the set of encrypted messages he receives to compute H_r in Step 2. The simulator $S_2^{\text{ppsvm}}(H_b, c_2)$ to simulate $VIEW_2^{\text{ppsvm}}$ is created as follows. Each message $E[m_b] \in \mathcal{V}_2^1$ is a Paillier encryption. To simulate it, S_2^{ppsvm} selects a random value r_b and computes $E[r_b]$. Because the Paillier encryption is semantically secure, $E[m_b]$ and $E[r_b]$ are computationally indistinguishable. The simulation of $\mathcal{V}_2^2$ is already given in Theorem 4.1. Thus, S_2^{ppsvm} can call the simulator in Theorem 4.1 to simulate $\mathcal{V}_2^2$.

We can extend the PPSVM to support more than two parties in computing the Gram matrix $\mathcal{G}$ as depicted in Algorithm 2. We assume k parties that are semi-honest but curious - they strictly follow the protocol and will not collude with each other. At the end of the protocol, each party P_i gets the partial result c_i of the $\mathcal{G}$ where $i \in \{1, \cdots, k\}$ and $\sum_{i=1}^{k} c_i = H \cdot H^T = \mathcal{G}$. The extended PPSVM works in a similar way as in the PPSVM (Algorithm 1). Thus, we omit the details here.

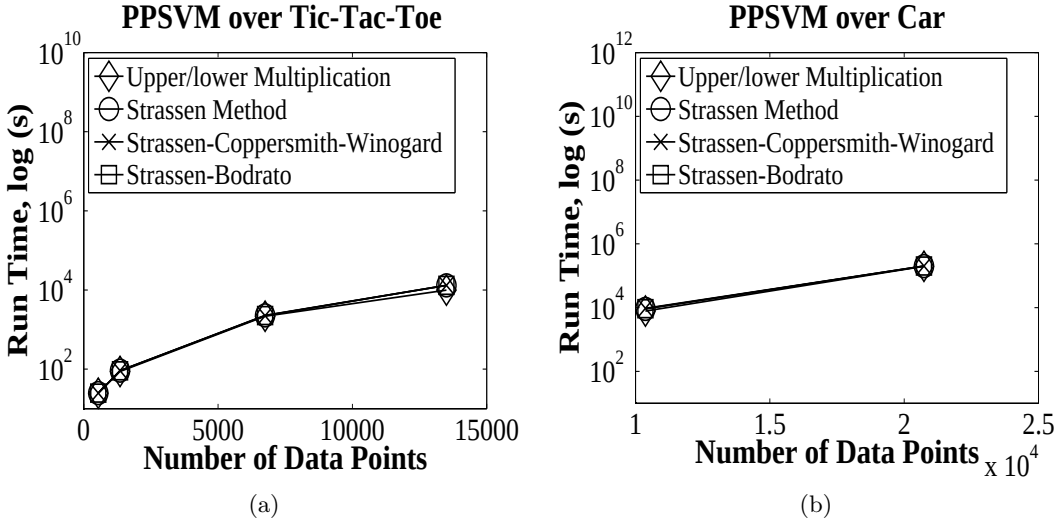
5.1.4 Experiment and Discussion

In this section, we evaluate the performance of our proposed privacy-preserving Support Vector Machine (PPSVM).

Dataset. We use two datasets for PPSVM. The first dataset is *Tic-Tac-Toe* [1] that contains 958 tuples with 27 predictor variables. Thus, the dataset contains 25866 (i.e., 968×27) data points. In this experiment, we evaluate *Tic-Tac-Toe* varying data points between 540 and 13500. The second is *Car* [2] that contains 3456 tuples with 6 predictor variables which is equivalent to 20736 data points. As our PPSVM can support only numerical predictors, we need to transform data of the categorical predictors of the *Car* into numerical values. Taking one categorical predictor (i.e., lug_boot) of the *Car* with values, "small", "med", and "big", we can convert the values into numerical values as follows: "small" $\rightarrow$ "01", "med" $\rightarrow$ "10", and "big" $\rightarrow$ "11".

We assume that Alice and Bob are semi-honest but curious in this experiment. All data points of the two datasets are split arbitrarily between Alice and Bob. Alice and Bob apply PPSVM to compute the Gram matrix $\mathcal{G}$ of the Support Vector Machine. At the end of PPSVM, Alice learns the partial result c_1 of $\mathcal{G}$ and nothing from Bob, and Bob learns the partial result c_2 of $\mathcal{G}$ and nothing from Alice, where $c_1 + c_2 = \mathcal{G}$.

Implementation. We implemented the solution by Java. All experiments were run on a machine Intel CPU 2.7GHz Core 7 processor with 4G memory running on Windows 7. We downloaded SMO package[1], and configured it to 1024-bit of Paillier homomorphic cryptosystem (Paillier, 1999). We set $\lambda = 40$ for secure multiplication of DAG model in PPSVM.



(a) (b)

Figure 5.2: Performance of the PPSVM

[1] http://www.datalab.uci.edu/people/xge/svm/
[2] http://archive.ics.uci.edu/ml/datasets/Car+Evaluation

The Evaluation. In Figures 5.2(a) and 5.2(b), running time increases as the number of data points increases in both *Tic-Tac-Toe* and *Car* datasets. The upper-lower matrix multiplication method runs faster than other three methods in both datasets. In 13500 data points of *Tic-Tac-Toe* , average run time per data point of the upper-lower matrix multiplication method is 0.7366 sec. In 20736 data points of *Car* , average run time per data point of the upper-lower matrix multiplication method is 9.5317 sec. The running times of the other three methods in both datasets are very close to each other. Our proposed PPSVM gets the gram matrix of each dataset that is identical to $\mathcal{G}$ (Gram matrix) computed by support vector machine without protecting data of Alice and that of Bob. The experiment results show that PPSVM integrated with DAG model is efficient and effective to securely compute $\mathcal{G}$ of which data is split arbitrarily between Alice and Bob.

5.2 Kernel Regression

A class of regression techniques (Friedman et al., 2001) can estimate the regression function $f(x)$ over the domain $\mathbb{R}^P$ by applying a simple model separately at each query point x_0. The model is fitted by observations close to the target point x_0. As a result, the estimated function $\hat{f}(x)$ is smooth in $\mathbb{R}^P$. To achieve the localization, a weighting function or kernel $K_h(x_0, x_i)$ assigning a weight to x_i based on the distance between x_0 and x_i can be applied. The width of the kernel, h, is the width of the neighborhood. Thus, kernel regression (i.e., kernel smoother) can apply kernel-based techniques for density estimation and classification.

We discuss the kernel regression algorithm in Section 5.2.1. In Section 5.2.2, we propose privacy-preserving kernel regression (PPKR) by DAG to securely compute the estimated function $\hat{f}(x)$. We discuss the security analysis and the complexity analysis of PPKR in Section 5.2.3. Lastly, we evaluate the performance of PPKR in Section 5.2.4.

Algorithm 2: Privacy-Preserving Multi-party Support Vector Machine (PPSVM) on arbitrarily partitioned data

Input: Matrix $H = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix}$ has $n \cdot n$ data points that are split arbitrarily among k parties. Let H_i be some data points of H for Party P_i where $i \in \{1, \cdots, k\}$ and $\sum_{i=1}^{k} H_i = H$.

Output: Party P_i gets c_i where $i \in \{1, \cdots, k\}$ and $\sum_{i=1}^{k} c_i = H \cdot H^T = \mathcal{G}$ (Gram matrix).

Party P_1 generates a pair of keys (sk, pk) (i.e., (secret key, public key)).
P_1 sends pk to other parties P_j where $j \in \{2, \cdots, k\}$.
P_i generates zero matrices $(n \times n)$, O_i, where $i \in \{1, \cdots, k\}$
P_1 adds O_1 into H_1 to get $H'_1 = H_1 + O_1$.
for *i = 1 to n* **do**
    for *j = 1 to n* **do**
        P_1 encrypts $E[d_{ij}^{P_1}]$ of H'_1 which is sent to P_2.
    end
end
Let U be an encrypted zero matrix.
for *m = 2 to k-1* **do**
    P_m adds O_m into H_m to get $H'_m = H_m + O_m$, and generates a random matrix R_m.
    for *i = 1 to n* **do**
        for *j = 1 to n* **do**
            P_m encrypts $Q_{ij} = E[d_{ij}^{P_{m-1}}] \times E[d_{ij}^{P_m}]$ and $U = \frac{U}{E[R_m]}$, which are sent to P_{m+1}.
        end
    end
    P_m sets $c_m = R_m$.
end
P_k adds O_k into H_k to get $H'_k = H_k + O_k$, and generates a random matrix R_k.
for *i = 1 to n* **do**
    for *j = 1 to n* **do**
        P_k encrypts $Q_{ij} = E[d_{ij}^{P_{k-1}}] \times E[d_{ij}^{P_k} + r_{k_{ij}}]$ which is sent to P_1.
    end
end
P_k also generates the second random matrix R'_k to compute $H_k = \frac{E[R_k^2] \times U}{(Q^T)^{R_k} \times Q^{R_k} \times E[R'_k]}$, which is sent to P_1. P_k sets $c_k = R'_k$.
P_1 decrypts $D[Q]$ and $D[H_k]$ to compute $c_1 = D[Q] \cdot (D[Q])^T + D[H_k]$ where $\sum_{i=1}^{k} c_i = H \cdot H^T = \mathcal{G}$.

5.2.1 Algorithm

Given a set of weights $\forall_{i=1}^{n} W_i(x)$ for each x as follows,

$$\hat{f}(x) = \sum_{i=1}^{n} W_i(x) y_i. \tag{5.9}$$

The weight function $W_i(x)$ is a density function that has a scale parameter to adjust the form and the size of the weights near to x. Given a scale parameter h, the weight sequence is

$$W_{hi}(x) = \frac{K_h(x, x_i)}{\sum_{i=1}^{n} K_h(x, x_i)}, \tag{5.10}$$

where

$$K_h(x, x_i) = D\left(\frac{|x - x_0|}{h}\right), \tag{5.11}$$

and $\sum_{i=1}^{n} W_{hi}(x_i) = 1$. For a simplicity, we write K_h as K in the following. For any x, the kernel regression can be plugging Equation 5.10 into Equation 5.9,

$$\hat{f}(x) = \sum_{i=1}^{n} W_{hi}(x) Y_i = \frac{\sum_{i=1}^{n} K(x, x_i) y_i}{\sum_{i=1}^{n} K(x, x_i)}. \tag{5.12}$$

In kernel regression, various Nadaraya-Watson kernel functions such as Gaussian kernel and Epanechnikov kernel, can be used as the kernel function K. The *Gaussian* kernel is defined as follows:

$$\mathsf{K}(x_i, x) = e^{-\mathsf{D}(x_i, x)/2h^2}, \tag{5.13}$$

where $\mathsf{D}(.)$ is the squared Euclidean distance, and $h > 0$ is the bandwidth controlling the smoothing. The *Epanechnikov* kernel is defined as follows:

$$K(x_i, x) = \begin{cases} \frac{3}{4}(1 - \frac{\mathsf{D}(x_i, x)}{2h^2}), & \text{if } \frac{\mathsf{D}(x_i, x)}{2h^2} < 1 \\ 0, & \text{otherwise.} \end{cases} \tag{5.14}$$

The scale (smoothing) parameter h is the width of the local neighborhood. In practice, larger h can lead to lower variance but higher bias. In the *Gaussian* kernel, h is the

Algorithm 3: Privacy-Preserving Kernel Regression (PPSVM) on horizontally partitioned data

Input: The training dataset has p tuples that each tuple has m predictors (i.e., $x'_1, \cdots, x'_m$) and 1 response value, y'. The p tuples are split horizontally between Alice and Bob. As a result, Alice has n tuples and Bob has l tuples, where $n + l = p$. Another dataset has q tuples (for testing known by Alice and Bob) that each tuple has m predictors (i.e., $x_1, \cdots, x_m$) without a response value.

Output: Alice gets c_1^i and Bob gets c_2^i where $i \in \{1, \cdots, q\}$, $\forall_{i=1}^{q} c_1^i + c_2^i \approx \hat{y}_i$, and $\hat{y}_i = \hat{f}(x)$ (Equation 5.12).

for *t = 1 to q* **do**
  Alice sets $a_1 = 0$ and $a_2 = 0$, and Bob sets $b_1 = 0$ and $b_2 = 0$.
  for *j = 1 to n* **do**
    Alice computes $S_a = \sum_{i=1}^{m} K((x_i)_t, (x'_i)_j)$, and then $a_2 = a_2 + S_a$ and $a_1 = a_1 + S_a \cdot (y')_j$.
  end
  for *g = 1 to l* **do**
    Bob computes $S_b = \sum_{i=1}^{m} K((x_i)_t, (x'_i)_g)$, and then $b_2 = b_2 + S_b$ and $b_1 = b_1 + S_b \cdot (y')_g$.
  end
  Alice and Bob jointly compute $\frac{a_1+b_1}{a_2+b_2} \approx c_1^t + c_2^t$, held by Alice and Bob respectively, where $c_1^t + c_2^t \approx \hat{y}_t$.
end

standard deviation. In contrast, h is measured by the radius of the support region in the *Epanechnikov* kernel. Moreover, the *Epanechnikov* kernel can provide a compact support in the nearest-neighbor window size.

5.2.2 Privacy-Preserving Kernel Regression (PPKR)

We propose privacy-preserving Kernel Regression (PPKR) that can securely compute the estimated function $\hat{f}(x)$ (Equation 5.9). PPKR can apply secure division operator of DAG to compute $\hat{f}(x)$ as depicted in Algorithm 3. We assume two parties, Alice and Bob, in our PPKR, that are semi-honest but curious - they strictly follow the protocol and will not collude with each other.

Let p be tuples in the training dataset. The p tuples are split horizontally between Alice and Bob. Alice and Bob have n and l tuples respectively, that each tuple has m predictors $(x'_1, \cdots, x'_m)$ with a response value y' where $n + l = p$. Another dataset (i.e., testing data) comes with q tuples that each tuple has m predictors $(x_1, \cdots, x_m)$ without a response value. The q tuples are known by both Alice and Bob.

Alice and Bob would like to predict $\forall_{i=1}^{q}\hat{y}_i$ values of the testing dataset using the estimated function $\hat{f}(x)$. Let $K(.)$ be the kernel function. Alice and Bob first configure a $(2,2)$-threshold Paillier cryptosytem with the public and private key pair (pk, sk). Suppose that sk_A and sk_B are the secret values (of Alice and Bob), which combined can recover sk. Let $E[.]$ and $D[.]$ be the encryption and decryption functions corresponding to pk and (sk_A, sk_B), respectively. Alice and Bob apply PPKR to securely compute $\hat{f}(x)$ step by step as follows.

Step 1. Alice locally computes a_1 and a_2 using her n training tuples. In each testing tuple of the $(x_i, \cdots, x_m)$ predictors, a_1 and a_2 are computed as

$$a_1 = \sum_{j=1}^{n}\sum_{i=1}^{m}(y')_j \cdot K((x_i)_t, (x'_i)_j), \qquad a_2 = \sum_{j=1}^{n}\sum_{i=1}^{m}K((x_i)_t, (x'_i)_j),$$

where $(y')_j$ is the response value of the j-th tuple, $(x'_i)_j$ is the i-th predictor in j-th tuple, and $(x_i)_t$ is the i-th predictor in t-th tuple of the q tuples. of the n tuples. Bob can locally compute b_1 and b_2 using his l training dataset. He computes b_1 (and b_2) in such a similar way of a_1 (and of that a_2) of Alice as follows.

$$b_1 = \sum_{g=1}^{l}\sum_{i=1}^{m}(y')_g \cdot K((x_i)_t, (x'_i)_g), \qquad b_2 = \sum_{g=1}^{l}\sum_{i=1}^{m}K((x_i)_t, (x'_i)_g),$$

where $(y')_g$ is the response value of the g-th tuple $(x'_i)_g$ is the i-th predictor in g-th tuple of the l tuples, and $(x_i)_t$ is the i-th predictor in t-th tuple of the q tuples.

Step 2. Alice and Bob jointly to compute $\hat{f}(x)$ which is equal to $\frac{a_1+b_1}{a_2+b_2}$. They apply secure division of DAG model to compute $\frac{a_1+b_1}{a_2+b_2}$.

Step 3. Alice gets c_1^t and Bob gets c_2^t where $c_1^t + c_2^t \approx \frac{a_1+b_1}{a_2+b_2} = \hat{y}_t$ (Equation 5.12) is the predicted value of t-th tuple in the testing dataset and $t \in \{1, \cdots, q\}$.

Alice and Bob can repeat above steps to compute a predicted value of each testing tuple. We discuss the security analysis and the complexity analysis of PPKR in the next section.

5.2.3 Security Analysis and Complexity Analysis

We use time complexity and communication complexity to measure the performance of privacy-preserving kernel regression (PPKR).

Time Complexity. We measure the time complexity of PPKR by modular exponentiations, since they consume most of the time. In Step 1, Alice and Bob locally compute a_1, a_2 and b_1, b_2 respectively, without involving any modular exponentiation. In Step 2, Alice and Bob need to call q times of secure divisions as the testing dataset contains q tuples. Thus, Step 2 takes $q(4\omega + 19 + 19z)$ modular exponentiations (refer to the time complexity of secure division in Section 4.1.5). Therefore, the number of modular exponentiations needed by PPKR is $q(4\omega + 19 + 19z)$ bounded by $O(q(\omega + z))$.

Communication Complexity. We measure the communication complexity by the number of message bits passing between Alice and Bob. In Step 1, Alice and Bob locally compute a_1, a_2 and b_1, b_2 respectively. In Step 2, Alice and Bob need to call q times of secure divisions as the testing dataset contains q tuples. Thus, Step 2 needs $q(t_2(24z+26+8w)+6(\chi+z+1)t_1 z)$ bits (refer to the communication complexity of secure division in Section 4.1.5). Therefore, the communication complexity is $q(t_2(24z+26+8w)+6(\chi+z+1)t_1 z)$ bits bounded by $O(zq(t_2 + t_1\chi) + t_2 q\omega)$.

Our proposed PPKR is proven secure via simulation paradigm (refer to Section 2.6.1 for more details) in the following.

Theorem 5.2 *The PPKR protocol is simulatable.*

Proof 5.2 *We simulate the view of Alice and that of Bob. We first simulate the view of Alice. Let (a_1, a_2) be the inputs of Alice, and c_1 be the protocol output to her. The view of Alice is $VIEW_1^{\text{ppkr}} = ((a_1, a_2), \mathcal{V}_1)$ where $\mathcal{V}_1$ is the set of messages she receives to compute $\hat{f}(x) = \frac{a_1+b_1}{a_2+b_2}$ in Step 2. The simulator $S_1^{\text{ppkr}}((a_1, a_2), c_1)$ to simulate $VIEW_1^{\text{ppkr}}$ is created as follows. The simulation of $\mathcal{V}_1$ is already given in Theorem 4.3. Thus, S_1^{ppkr} can call the simulator in Theorem 4.3 to simulate $\mathcal{V}_1$.*

The view of Bob is simulated in the following. Let (b_1, b_2) be the inputs of Bob, and c_2 be the protocol output to him. The view of Bob is $VIEW_2^{\text{ppkr}} = ((b_1, b_2), \mathcal{V}_2)$, where $\mathcal{V}_2$ is the set of messages he receives from Alice in Step 2 to compute $\hat{f}(x) = \frac{a_1+b_1}{a_2+b_2}$. The simulator $S_2^{\text{ppkr}}((b_1, b_2), c_2)$ to simulate $VIEW_2^{\text{ppkr}}$ is created as follows. The simulation of $\mathcal{V}_2$ is already given in Theorem 4.3. Thus, S_2^{ppkr} can call the simulator in Theorem 4.3 to simulate $\mathcal{V}_2$.

5.2.4 Experiment and Discussion

In this section, we evaluate the performance of our proposed privacy-preserving kernel regression (PPKR).

Dataset. We use two datasets for PPKR. The first is *(Red) Wine* [3] to predict the quality of red wine. It contains 1,598 tuples. The second is *(Consumption) Power* [4] to predict household power consumption. After removing tuples with missing values, it contains 2,049,280 tuples. For each dataset, we randomly select 1/3 tuples as the training data for Alice, a second 1/3 tuples as the training data for Bob, and randomly select 500 tuples from the remaining 1/3 tuples as testing data. We scale the values of each dimension to $[0.0, 1.0]$.

Table 5.1: Experiment Parameters for PPKR

Operator	κ	γ	β	τ	w	z	λ
/	950	15	12	28	20	-	40
bit-length	-	-	-	-	-	45	

Implementation. We implemented the solutions by Java. All experiments were run on a machine of Intel® Xeon® 2.3GHz CPU (16 cores) with 128.0GB RAM running on Windows Server 2012. We downloaded the threshold Paillier Cryptosystem [5], and configured it to 1024-bit. We use CMP of the fast garbled circuit [6] that is one of the most efficient implementations for secure comparison. Table 5.1 gives the values of the security parameters we use.

In PPKR, we assume that Alice and Bob would like to predict the $\forall_{t=1}^{q}\hat{y}_t$ values for the testing data, but are not willing to disclose their training data for privacy protection. The secure division of DAG model (Section 4.1.5) can be used to predict [7] the $\hat{f}(x) = \hat{y}_t$ value in Equation 5.12. PPKR is detailed in Section 5.2.2. In the experiments, we use two Nadaraya-Watson kernel functions. The first is *Gaussian* kernel in Equation 5.13. The second is *Epanechnikov* kernel in Equation 5.14.

[3] http://archive.ics.uci.edu/ml/datasets/Wine+Quality
[4] http://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption
[5] http://www.utdallas.edu/∼mxk093120/paillier/
[6] https://github.com/yhuang912/FastGC
[7] Alice and Bob release the protocol output c_1 and c_2.

We use Mean Squared Error (MSE) to measure the difference between the real y and the predicted value $\hat{y}$,

$$\text{MSE}(\hat{y}) = \frac{1}{q}\sum_{i=1}^{q}(\hat{y}_i - y_i)^2, \tag{5.15}$$

where q is the number of testing tuples.

Our secure operators proposed in Section 4.1 have security parameters. We assume that values (i.e., a_1, a_2, b_1, b_2) are in the range of $[2^{-15}, 2^{15}]$ (i.e., $\gamma = 15$ in Lemma 4.2). If they are beyond the range, we truncate them to their nearest bounds. Table 5.1 gives the parameters and their values used in PPKR.

The Evaluation. We first study the kernel regression by Gaussian kernel (Equation 5.13) on the Wine and Power datasets, which are configured at the beginning of Section 5.2.4. We fix the number of predictor variables to 6, and vary the bandwidth h. Figure 5.3(a) reports the MSE. *Wine-P* and *Power-P* are the results of private setting, where our DAG model is applied on datasets Wine and Power, respectively. *Wine-NP* and *Power-NP* are the results of non-private setting, where Alice and Bob disclose their data directly. The results of private and non-private setting are very close. Even in the worst case of *Wine-P* and *Wine-NP* at $h = 0.1$, the difference is small – the MSE of *Wine-NP* is 0.0247 and that of *Wine-P* 0.0249. Figure 5.3(b) shows the running time. As the h value increases, the running time does not change obviously. On the smaller Wine dataset, the running time for one prediction of non-private setting is 0.008 sec, and that of private setting is 8.7647 sec. On the much bigger Power dataset, the running time for one prediction of non-private setting is 5.9347 sec, and that of private setting is 12.3480 sec. The prediction time is higher for bigger dataset. This is as expected, since kernel regression requires to compute the kernel function on each tuple in the training dataset.

We then fix the bandwidth to 0.1 and study the effect of number of predictor variables. Figure 5.3(c) is the results of MSE as the number of predictor variables increases from 1 to 6. Clearly, the third predictor variable is very useful for the prediction of the Power dataset; the MSE decreases significantly when the number of predictor variables increases from 2 to 3. Again, the MSE of private setting and that of non-private setting are very close. Figure 5.3(d) gives the running time. As the number of predictor variables increases, the kernel evaluation takes more time. Thus, running time for all the cases increases.

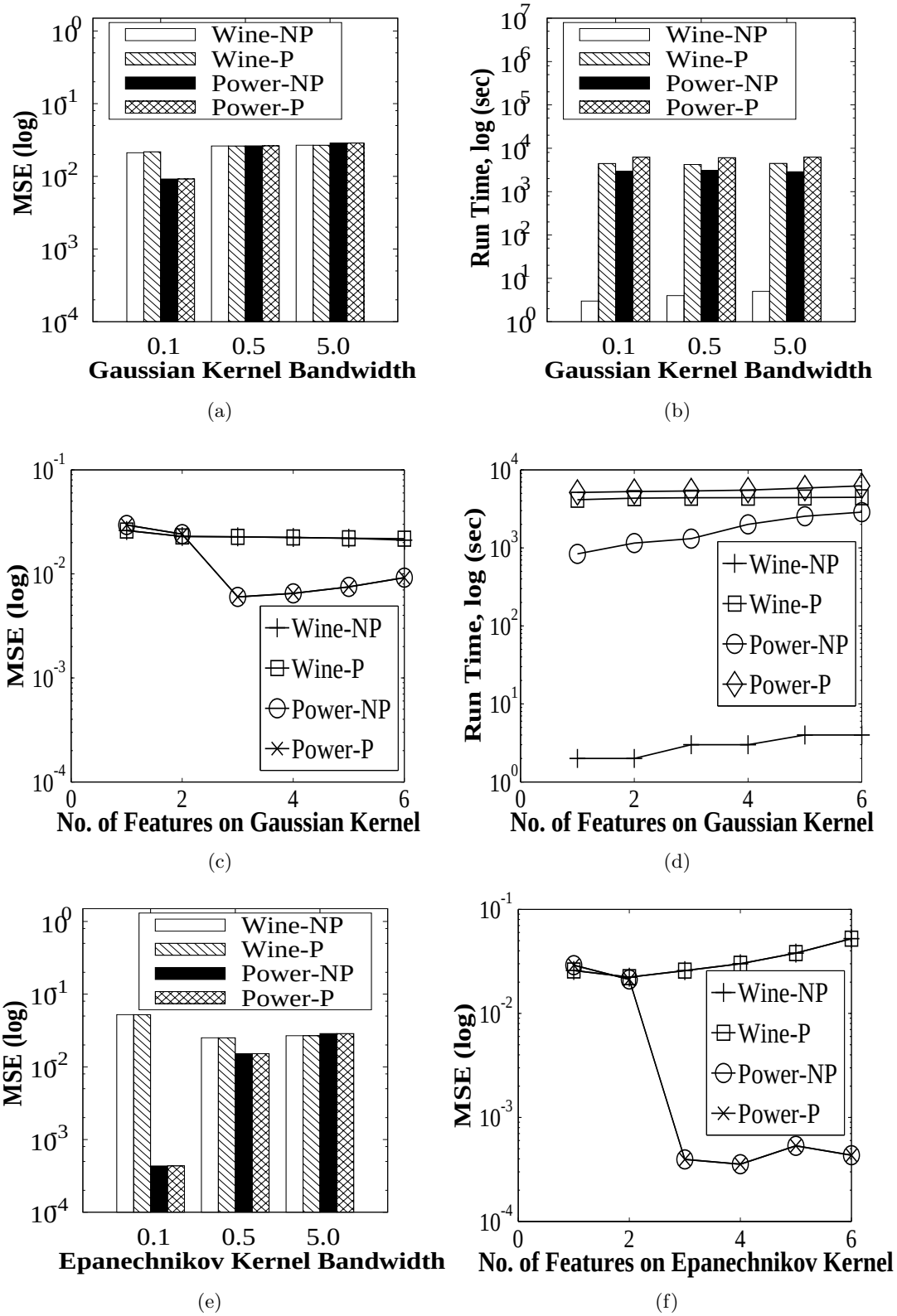


Figure 5.3: Kernel Regression with/without Privacy Preservation

We now evaluate the kernel regression using Epanechikov kernel (Equation 5.14). Figure 5.3(e) gives the experimental results of varying bandwidth. Comparing Figure 5.3(e) with Figure 5.3(a), we can see that the effect of bandwidth on different kernels is different

– on Gaussian kernel $h = 0.1$ gives the best results, while on Epanechikov kernel $h = 0.5$ is preferred. Figure 5.3(f) varies the number of predictor variables. The running time of Epanechikov kernel is similar to that of Gaussian kernel. We omit its report here.

5.3 Naïve Bayes

A Bayesian classifier (Mitchell, 1997) is a statistical classifier based on Bayes theorem. It can predict probabilities of class members; e.g., given a sample of a dataset, the Bayesian classifier can calculate the probability of the sample that belongs to a particular class. To reduce the learning complexity in Bayesian classifier, Naïve Bayes assumes all predictors (i.e., features) are conditionally independent. (Domingos and Pazzani, 1996) show that the Naïve Bayesian learning is effective in comparable to performance with other classifiers such as neural network and decision tree.

In the following, the Naïve Bayes algorithm is discussed in Section 5.3.1. In Section 5.3.2, we propose privacy-preserving Naïve Bayes (PPNB) by DAG to securely build a model and then evaluate the model. We discuss the security analysis and the complexity analysis of PPNB in Section 5.3.3. Lastly, we evaluate the performance of PPNB in Section 5.3.4.

5.3.1 Algorithm

In this section, we briefly discuss the Naïve Bayes classifier (Mitchell, 1997). Let $A_1, A_2, \cdots, A_n$ be predictors that are conditionally independent with each other, given C. Thus, based on Bayes theorem, we can write it as

$$\begin{aligned} P(A|C) =& P(A_1, A_2, \cdots, A_n|C) \\ =& P(A_1|C)P(A_2|C)\cdots P(A_n|C) = \prod_{i=1}^{n} P(A_i|C). \end{aligned} \tag{5.16}$$

Next, we assume that (in general) C is a discrete variable and $A_1, A_2, \cdots, A_n$ are nominal or continuous predictor variables. Let $m_1, m_2, \cdots, m_k$ be values of C. Given a new instance A, the Naïve Bayes classifier can compute the probability C taking $m_i \in 1, \cdots, k$

as follows,

$$P(C = m_i|A_1, A_2, \cdots, A_n) = \frac{P(C = m_i)P(A_1, A_2, \cdots, A_n|C = m_i)}{\sum_{j=1}^{k} P(C = m_j)P(A_1, A_2, \cdots, A_n|C = m_j)}, \quad (5.17)$$

where the sum is added by each probability of all possible values of C. If $A_1, A_2, \cdots, A_n$ are conditional independent given C, we can substitute Equation 5.16 into 5.17 as

$$P(C = m_i|A_1, A_2, \cdots, A_n) = \frac{P(C = m_i)\prod_{i=1}^{n} P(A_i|C = m_i)}{\sum_{j=1}^{k} P(C = m_j)\prod_{i=1}^{n} P(A_i|C = m_j)}. \quad (5.18)$$

Thus, the probability C that takes any value can be computed as the observed predictor values of a new instance and the distributions $P(C)$ and $P(A_i|C)$ estimated from training data are given. Most probable value of C can find by

$$C \leftarrow \arg\max_{m_i} \frac{P(C = m_i)\prod_{i=1}^{n} P(A_i|C = m_i)}{\sum_{j=1}^{k} P(C = m_j)\prod_{i=1}^{n} P(A_i|C = m_j)}, \quad (5.19)$$

which can simplify to

$$C \leftarrow \arg\max_{m_i} P(C = m_i)\prod_{i=1}^{n} P(A_i|C = m_i). \quad (5.20)$$

More details of Naïve Bayes can be found in (Mitchell, 1997).

5.3.2 Privacy-Preserving Naïve Bayes (PPNB)

We propose privacy-preserving Naïve Bayes (PPNB) that can securely build a classifier model and then use it to test instances (i.e., tuples). PPNB can apply secure operators of DAG to perform tasks above. We assume two parties, Alice and Bob, in our PPNB, that are semi-honest but curious - they strictly follow the protocol and will not collude with each other.

In the PPNB, a training dataset is used to build the model. The model can be used to evaluate tuples of a testing dataset (i.e., testing data). Both the training dataset and the testing dataset contain the same number of predictors. In the two-party setting, the training dataset is split horizontally between Alice and Bob. Thus, PPNB needs to

use tuples of Alice and that of Bob in constructing the model without revealing their tuples to each other. Alice (or Bob) can then use the model to evaluate the tuples of the testing dataset. We next discuss two tasks of the PPNB: building the classifier model and evaluating the tuples of the testing dataset.

Building the Classifier Model. The classifier model is built from nominal and continuous predictors of the training dataset split between Alice and Bob. In the end of the construction, the model is split into the two portions: the portion of Alice and the portion of Bob. In Naïve Bayes, the attributes are conditionally independent with each other given the response value (i.e., target value). Let $(x_1, x_2, \cdots, x_m, y)$ be data format of the training dataset where x_i's for $i = 1, 2, \cdots, m$ are predictor variables and y is the response variable. There are two types of the predictor in PPNB: the nominal predictor and the continuous predictor.

Nominal Predictor. We first discuss to compute the probability of a value v_{ji} of the nominal predictor x_j with a response value y_k step by step as follows. Alice and Bob first configure a $(2, 2)$-threshold Paillier cryptosytem with the public and private key pair (pk, sk). Suppose that sk_A and sk_B are the secret values (of Alice and Bob), which combined can recover sk. Let $E[.]$ and $D[.]$ be the encryption and decryption functions corresponding to pk and (sk_A, sk_B), respectively. Let r and l be the number of response values and the number of nominal predictors, respectively. The training dataset contains p tuples split between Alice and Bob, such that Alice has g tuples and Bob has h tuples where $g + h = p$.

Step 1. Alice locally computes N^A_{yk} which is the number of records that have response value y_k in g tuples of Alice. Bob locally computes $N^B_{y_k}$ which is the number of records that have response value y_k in h tuples of Alice.

Step 2. Alice locally computes $N^A_{x_j=v_{ji},y_k}$ which is the number of records of the nominal predictor x_j that has the value v_{ji} with the response value y_k of the g tuples. Bob also locally computes $N^B_{x_j=v_{ji},y_k}$ which is the number of records of the nominal predictor x_j that has the value v_{ji} with the response value y_k of the h tuples. To avoid 0 (in both cases, $N^A_{x_j=v_{ji},y_k}$ and $N^B_{x_j=v_{ji},y_k}$) in probability, Alice and Bob can apply the Laplace smoothing (Mitchell, 1997) to the value v_{ji} of the predictor x_j.

Algorithm 4: Nominal Predictors in PPNB

Input: Let r and l be the number of response values and the number of nonimal predictors respectively, in the training dataset. For simplicity, we assume that each nominal predictor has q values. The training dataset contains p tuples split horizontally between Alice and Bob, such that Alice has g tuples and Bob has h tuples where $g + h = p$.

Output: Alice gets $c_1^{x_j=v_{ji},y_k}$ and Bob gets $c_2^{x_j=v_{ji},y_k}$ where $\forall_{k=1}^{r}\forall_{j=1}^{l}\forall_{i=1}^{q}\left(c_1^{x_j=v_{ji},y_k} + c_2^{x_j=v_{ji},y_k}\right) = \mathsf{Pr}\left(x_j = v_{ji}|y_k\right)$.

```
for k = 1 to r do
    Alice computes N^A_{y_k} which is the number of records that have response value y_k
    in g tuples of Alice.
    Bob computes N^B_{y_k} which is the number of records that have response value y_k
    in h tuples of Bob.
    for j = 1 to l do
        for i = 1 to q do
            Alice computes N^A_{x_j=v_ji,y_k} which is the number of records of the nominal
            predictor x_j that has the value v_ji with the response value y_k in g tuples
            of Alice.
            Bob computes N^B_{x_j=v_ji,y_k} which is the number of records of the nominal
            predictor x_j that has the value v_ji with the response value y_k in h tuples
            of Bob.
            Alice and Bob jointly compute (N^A_{x_j=v_ji,y_k} + N^B_{x_j=v_ji,y_k}) / (N^A_{y_k} + N^B_{y_k})
            using secure division of DAG model. The outputs are c_1^{x_j=v_ji,y_k} and
            c_2^{x_j=v_ji,y_k}, held by Alice and Bob respectively, where
            c_1^{x_j=v_ji,y_k} + c_2^{x_j=v_ji,y_k} = Pr(x_j = v_ji|y_k).
        end
    end
end
```

Step 3. Alice and Bob jointly compute $\frac{N^A_{x_j=v_{ji},y_k}+N^B_{x_j=v_{ji},y_k}}{N^A_{y_k}+N^B_{y_k}}$ using secure division of DAG model. The outputs are $c_1^{x_j=v_{ji},y_k}$ and $c_2^{x_j=v_{ji},y_k}$, held by Alice and Bob respectively, where $c_1^{x_j=v_{ji},y_k} + c_2^{x_j=v_{ji},y_k} = \mathsf{Pr}\left(x_j = v_{ji}|y_k\right)$.

Alice and Bob can repeat above steps (1-3) to securely compute the probabilities of the nominal predictors. The probability calculation of nominal predictors is detailed in Algorithm 4.

Continuous Predictor. In the model construction, Alice and Bob need to compute only the mean u and the standard deviation σ of the continuous predictor x_j with the response value y_k. Again, we can apply the same setting of the (2,2)-threshold Paillier cryptosystem of the probability calculation of the nominal predictors. Let r and ℓ be the number of response values and the number of continuous predictors, respectively.The training dataset contains p tuples split between Alice and Bob, such that Alice has g tuples and Bob has

h tuples where $g + h = p$. Alice and Bob can compute u and σ of the predictor x_j with the response value y_k step by step as follows,

Step 1. Alice locally computes N^A_{yk} which is the number of records that have response value y_k in g tuples of Alice. Bob locally computes $N^B_{y_k}$ which is the number of records that have response value y_k in h tuples of Bob.

Step 2. Alice locally computes $S^A_{x_j,y_k}$ which is the summation of the values of the continuous predictor x_j with the response value y_k of g tuples. Bob also computes $S^B_{x_j,y_k}$ which is the summation of the values of the continuous predictor x_j with the response value y_k of h tuples.

Step 3. Alice and Bob jointly compute $\frac{S^A_{x_j,y_k}+S^B_{x_j,y_k}}{N^A_{yk}+N^B_{yk}}$, using secure division of DAG model. The outputs are $\mu_1^{x_j,y_k}+\mu_2^{x_j,y_k}$, held by Alice and Bob, respectively, where $\mu_1^{x_j,y_k}+\mu_2^{x_j,y_k} = \mu^{x_j,y_k}$ is the mean of the predictor x_j with the response value y_k.

Step 4. Alice and Bob can apply secure multiplication to jointly compute

$$\begin{aligned}(\mu_1^{x_j,y_k} + \mu_2^{x_j,y_k})^2 &= (\mu_1^{x_j,y_k})^2 + 2(\mu_1^{x_j,y_k})(\mu_2^{x_j,y_k}) + (\mu_2^{x_j,y_k})^2 \\ &= \mu_1^{x_j,y'_k} + \mu_2^{x_j,y'_k}, \end{aligned} \tag{5.21}$$

where $\mu_1^{x_j,y'_k}$ and $\mu_2^{x_j,y'_k}$ are held by Alice and Bob respectively. Alice then uses secure multiplication to compute $\sum_{v\in V^A_{y_k}}(v \cdot x_j - \mu^{x_j,y_k})^2$ with the outputs of of Equation 5.21. where v is a record in the subset and $v \cdot x_j$ is the value v of predictor x_j in g tuples. Likewise Bob uses secure multiplication to computes $\sum_{v\in V^B_{y_k}}(v \cdot x_j - \mu^{x_j,y_k})^2$ where v is a record in the subset and $v \cdot x_j$ is the value v of predictor x_j in h tuples. Alice and Bob next apply secure division to jointly compute

$$\begin{aligned}(\sigma^{x_j,y_k})^2 &= \left(\frac{\sum_{v\in V^A_{y_k}}(v \cdot x_j - \mu^{x_j,y_k})^2 + \sum_{v\in V^B_{y_k}}(v \cdot x_j - \mu^{x_j,y_k})^2}{N^A_{y_k} + N^B_{y_k} - 1}\right) \\ &= (\sigma_1^{x_j,y_k})_s + (\sigma_2^{x_j,y_k})_s, \end{aligned} \tag{5.22}$$

where $(\sigma_1^{x_j,y_k})_s$ and $(\sigma_2^{x_j,y_k})_s$, held by Alice and Bob respectively, are the square deviation $(\sigma^{x_j,y_k})^2$ of the predictor x_j with the response value y_k. To get the standard deviation σ,

Algorithm 5: Continuous Predictors in PPNB

Input: Let r and ℓ be the number of response values and the number of continuous predictors respectively, in the training dataset. The training dataset contains p tuples split horizontally between Alice and Bob, such that Alice has g tuples and Bob has h tuples where $g + h = p$.

Output: Alice gets $\mu_1^{x_j,y_k}, \sigma_1^{x_j,y_k}$ and Bob gets $\mu_2^{x_j,y_k}, \sigma_2^{x_j,y_k}$ where $\forall_{k=1}^{r}\forall_{j=1}^{l}\left(\mu_1^{x_j,y_k} + \mu_2^{x_j,y_k}\right) = \mu^{x_j,y_k}$ and $\forall_{k=1}^{r}\forall_{j=1}^{l}\left(\sigma_1^{x_j,y_k} + \sigma_2^{x_j,y_k}\right) = \sigma^{x_j,y_k}$.

1 **for** *k = 1 to r* **do**

2 Alice computes $N_{y_k}^A$ which is the number of records that have response value y_k in g tuples of Alice.

3 Bob computes $N_{y_k}^B$ which is the number of records that have response value y_k in h tuples of Bob.

4 **for** *j = 1 to ℓ* **do**

5 Alice computes S_{x_j,y_k}^A which is the summation of the values of the continuous predictor x_j with the response value y_k in g tuples of Alice.

6 Bob computes S_{x_j,y_k}^B which is the summation of the values of the continuous predictor x_j with the response value y_k in h tuples of Bob.

7 Alice and Bob jointly compute $\frac{S_{x_j,y_k}^A + S_{x_j,y_k}^B}{N_{y_k}^A + N_{y_k}^B}$ using secure division of DAG model. The outputs are $\mu_1^{x_j,y_k} + \mu_2^{x_j,y_k}$, held by Alice and Bob respectively, where $\mu_1^{x_j,y_k} + \mu_2^{x_j,y_k} = \mu^{x_j,y_k}$ is the mean of the continuous predictor x_j with the response value y_k.

8 Alice and Bob jointly compute $\left(\frac{\sum_{v\in V_{y_k}^A}(v\cdot x_j - \mu^{x_j,y_k})^2 + \sum_{v\in V_{y_k}^B}(v\cdot x_j - \mu^{x_j,y_k})^2}{N_{y_k}^A + N_{y_k}^B - 1}\right)^{\frac{1}{2}}$ using secure multiplication, secure division and also secure power of DAG model, where v is a record in the subset and $v \cdot x_j$ is the value v of the predictor x_j. The outputs are $\sigma_1^{x_j,y_k} + \sigma_2^{x_j,y_k}$, held by Alice and Bob respectively, where $\sigma_1^{x_j,y_k} + \sigma_2^{x_j,y_k} = \sigma^{x_j,y_k}$ is the standard deviation of the continuous predictor x_j with the response value y_k.

9 **end**

10 **end**

Alice and Bob can apply secure power of DAG model,

$$\begin{aligned}(\sigma^{x_j,y_k}) &= ((\sigma^{x_j,y_k})^2)^{\frac{1}{2}} = ((\sigma_1^{x_j,y_k})_s + (\sigma_2^{x_j,y_k})_s)^{\frac{1}{2}} \\ &= \sigma_1^{x_j,y_k} + \sigma_2^{x_j,y_k}, \end{aligned} \tag{5.23}$$

where $\sigma_1^{x_j,y_k}$ and $\sigma_2^{x_j,y_k}$, held by Alice and Bob respectively, are the standard deviation σ^{x_j,y_k} of the predictor x_j with the response value y_k. The mean of the attribute x_j with the response value y_k is $\mu_1^{x_j,y_k}$ and $\mu_2^{x_j,y_k}$, held by Alice and Bob respectively.

Alice and Bob can repeat above steps (1-4) to compute the means and the standard deviations of the continuous predictors. The mean and the standard deviation calculations of continuous predictors are detailed in Algorithm 5.

Algorithm 6: Testing Tuples in PPNB

Input: Let r and m be the number of response values and the number of predictors , respectively. The testing dataset contains d tuples.

Output: Alice gets $\forall_{j=1}^{d} MAP_{j1}^{A}$ and Bob gets $\forall_{j=1}^{d} MAP_{j2}^{B}$ where $MAP_{j1}^{A} + MAP_{j2}^{B} = MAP_j$ is the maximum probability of the j-th tuple in the testing dataset.

for *j = 1 to d* **do**
  for *k = 1 to r* **do**
    Alice and Bob jointly compute probabilities of the continuous predictors using Equation 5.26.
    Alice and Bob jointly compute $\hat{y}_k = \mathsf{Pr}(y_k) \prod_{p=1}^{m} \mathsf{Pr}(x_p|y_k) = \hat{y}_{k1}^{A} + \hat{y}_{k2}^{B}$, held by Alice and Bob respectively, using secure multiplication of DAG model.
  end
  Alice and Bob apply secure max location of DAG model to get $\max(\hat{y}_{11}^{A} + \hat{y}_{12}^{B}, \cdots, \hat{y}_{r1}^{A} + \hat{y}_{r2}^{B}) = \hat{y}_{i1}^{A} + \hat{y}_{i2}^{B}$, held by Alice and Bob respectively, where $i \in \{1, \cdots, r\}$. Alice sets $MAP_{j1}^{A} = \hat{y}_{i1}^{A}$ and Bob sets $MAP_{j2}^{B} = \hat{y}_{i2}^{B}$.
end

The model also contains probabilities of the response values of the training dataset. Alice and Bob jointly compute the probability of the response value y_k in the following. Let g be tuples of Alice and h be tuples of Bob, where $g + h$ is the total tuples in the training dataset. Alice first locally computes $N_{y_k}^{A}$ which is the number of the response value y_k in g tuples of Alice. Bob also locally computes $N_{y_k}^{B}$ which is the number of the response value y_k in h tuples of Bob. Alice and Bob then apply secure division to compute

$$\mathsf{Pr}(y_k) = \frac{N_{y_k}^{A} + N_{y_k}^{B}}{g + h} = \mathsf{Pr}(y_k)^{A} + \mathsf{Pr}(y_k)^{B}, \tag{5.24}$$

where $\mathsf{Pr}(y_k)^{A}$ and $\mathsf{Pr}(y_k)^{B}$, held by Alice and Bob respectively, are the probability of the response value y_k. Alice and Bob can repeat above steps to compute the probabilities of other response values. At the end of model construction, Alice and Bob each also contain the partial results of the probabilities of the nominal predictors and that of the means and the standard deviations of the continuous predictors.

Evaluating the Testing Tuples. Alice (Bob) can predict response values for tuples in the testing dataset. Again, let $(x_1, x_2, \cdots, x_m)$ be data format of the testing dataset and $\hat{y}$ be the response value, where x_i's for $i = 1, 2, \cdots, m$ are predictor variables. We can

rewrite Equation 5.20 to predict $\hat{y}$ of the testing tuple as

$$\hat{y} = \arg\max_{y_k \in \Omega} \left(\mathsf{Pr}\left(y_k\right) \prod_{j=1}^{m} \mathsf{Pr}\left(x_j | y_k\right) \right), \tag{5.25}$$

where Ω is the domain of response variable. Alice and Bob can retrieve the probability $\mathsf{Pr}\left(y_k\right)$ of the response value y_k and the probability $\mathsf{Pr}\left(x_j | y_k\right)$ of the nominal predictor x_j with the response value y_k that all probabilities have been calculated in the model construction. However, Alice and Bob still need to compute the conditional probabilities of the continuous predictors. We can adopt the typical assumption (Han et al., 2006) that the probability density function (pdf) of a continuous predictor is a Gaussian distribution.

$$\mathsf{Pr}\left(x_j = v_{ji} | y_k\right) = \frac{1}{\sqrt{2\pi}\sigma^{x_j, y_k}} \exp\left(-\frac{(v_{ij} - \mu^{x_j, y_k})^2}{2(\sigma^{x_j, y_k})^2} \right), \tag{5.26}$$

where μ^{x_j, y_k} and σ^{x_j, y_k} are the mean and the standard deviation of the value v_{ji} of the continuous predictor x_j with the response value y_k. Clearly, μ^{x_j, y_k} and σ^{x_j, y_k} that have been calculated in the task of building the classifier model can be retrieved by Alice and Bob without any secure computation. They can apply secure multiplication and secure division of DAG model to compute the probability $\mathsf{Pr}\left(x_j = v_{ji} | y_k\right)$ of the value v_{ji} of the continuous predictor x_j with the response value y_k in Equation 5.26. As the probability computation is straightforward, we omit the details here.

Next, Alice (Bob) predicts the response value of the testing tuple. For a simplicity, we assume that the testing dataset contains r response values. Let $\hat{y}_i^A$ and $\hat{y}_i^B$, held by Alice and Bob respectively, be probability of a response value where $\hat{y}_i^A + \hat{y}_i^B = \hat{y}_i$ and $i \in \{1, \cdots, r\}$. In each testing tuple, Alice and Bob apply secure multiplication on Equation 5.25 to compute the list of the probabilities of the response values, $(\hat{y}_{11}^A + \hat{y}_{12}^B, \hat{y}_{21}^A + \hat{y}_{22}^B, \cdots, \hat{y}_{r1}^A + \hat{y}_{r2}^B)$. In the next step, Alice (Bob) can find the maximum probability of the list as follows.

$$MAP_j = \max(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_k) = \max(\hat{y}_{11}^A + \hat{y}_{12}^B, \hat{y}_{21}^A + \hat{y}_{22}^B, \cdots, \hat{y}_{r1}^A + \hat{y}_{r2}^B), \tag{5.27}$$

where $j \in \{1, \cdots, r\}$. To find the maximum response value in Equation 5.27, Alice and Bob can apply secure max location of our DAG model (Section 4.1.9). The secure max protocol only discloses the maximum probability of the list, MAP_j in Equation 5.27,

Table 5.2: Secure operators in model construction per dataset

	$\times$	$/$	power
np nominal predictors		$\sum_{i=1}^{np} \lvert X_i \rvert \cdot \lvert \Omega \rvert$	
cp continuous predictors	$2 \cdot \lvert \Omega \rvert \cdot cp$	$2 \cdot \lvert \Omega \rvert \cdot cp$	$\lvert \Omega \rvert \cdot cp$

$|X_i|$ is the domain size of i-th nominal predictor x_i.
$|\Omega|$ is the domain size of response variable y.

while the comparison result between $\hat{y}_1$ and $\hat{y}_2$ for $\hat{y}_1 \neq \hat{y}_2$ should be kept confidential. Thus, Alice (Bob) can predict the response values $\hat{y}_s$ of the testing tuples, as detailed in Algorithm 6. In the case that the testing tuple is given only two response values (i.e., two probabilities), Alice and Bob can apply CMP (Section 3.1.10) directly to know whether $\hat{y}_{11}^A + \hat{y}_{11}^B \geq \hat{y}_{21}^A + \hat{y}_{21}^B$ by checking if $\hat{y}_{11}^A - \hat{y}_{21}^A \geq \hat{y}_{21}^B - \hat{y}_{11}^B$ holds.

In PPNB, Alice and Bob can apply secure operators of DAG to perform the two tasks above, building the classifier model and evaluating the testing tuples, with privacy preservation of Alice's data and that of Bob's data. We will discuss the security and the complexity analysis in the next section.

5.3.3 Security Analysis and Complexity Analysis

We use time complexity and communication complexity to measure the performance of privacy-preserving Naïve Bayes (PPNB). Alice and Bob combine secure operators of DAG model to securely perform the two tasks in PPNB, building the classifier model and evaluating the testing tuples. Table 5.2 reports the number of secure operators to build a classifier model for each dataset. The operators needed are dependent on the number of nominal predictor variables (assumed to be np), the number of continuous predictor variables (assumed to be cp), and the domain sizes of nominal and response variables. The operators needed to predict the class label for testing data are also up to predictor and response variables. Table 5.3 summarizes the results.

Table 5.3: Secure operators in model testing per tuple

	$\times$	$/$
np nominal predictors	$\lvert \Omega \rvert \cdot np$	
cp continuous predictors	$4 \cdot \lvert \Omega \rvert \cdot cp$	$2 \cdot \lvert \Omega \rvert \cdot cp$

$|\Omega|$ is the domain size of response variable y.

Time Complexity. We measure the time complexity of PPNB by modular exponentiations, since they consume most of the time.

Table 5.4: Time complexities of secure operators

Secure operator	Time complexity	Asymptotic time
$\times$ (Section 4.1.3)	6	$O(1)$
/ (Section 4.1.5)	$4\omega + 46 + 36z$	$O(\omega + z)$
power (Section 4.1.7)	$5\omega + 45 + 36z$	$O(\omega + z)$
max location (Section 4.1.9)	$24l - 2$	$O(l)$

ω is the number of iterations in Taylor series.
z is the maximum bit-length of input data.
l is the number of probabilities in the list.

Based on Tables 5.2 and 5.3, the time complexity of PPNB can be easily computed using Table 5.4. Thus, we omit the details here.

Communication Complexity. We measure the communication complexity by the number of message bits passing between Alice and Bob. Based on Tables 5.2 and 5.3, the communication complexity of PPNB can be easily computed using Table 5.5. Thus, we omit the details here.

Table 5.5: Communication complexities of secure operators

Secure operator	Communication complexity	Asymptotic communication
$\times$	$2t_2$	$O(1)$
/	$t_2(24z + 24 + 8w) + 6(\lambda + z + 2)t_1 z$	$O(z(t_2 + t_1\lambda) + t_2\omega)$
max location	$16t_2 l - 4t_2 + 3(\phi)t_1(l - 1)$	$O(l(t_2 + t_1))$

t_1 is a security parameter and t_2 is the message length in Paillier cryptosystem.
ω is the number of iterations in Taylor series. λ is the threshold (e.g., $\lambda = 40$)
z is the maximum bit-length of input data. l is the number of probabilities in the list.

Our proposed PPNB is proven secure via simulation paradigm (refer to Section 2.6.1 for more details) in the following.

Theorem 5.3 *The PPNB protocol is simulatable.*

Proof 5.3 *In PPNB, Alice and Bob use combined operators of* DAG *model to perform the two tasks, building the classifier model and evaluating the testing tuples. Thus, the view of Alice and that of Bob can be simulated using a similar proof in Theorem 4.8. We omit the details here.*

5.3.4 Experiment and Discussion

In this section, we evaluate the performance of our proposed privacy-preserving Naïve Bayes (PPNB).

Dataset. We use two datasets for PPNB. The first is *Adult* [8] to predict whether a person's salary is above 50,000 or not (i.e., 2 response values). It has 14 nominal predictors . We remove 2 of them: capital gain and capital loss, and keep the remaining 12. The second is *Mushroom* [9] to predict edible mushrooms (i.e., 2 response values) using 22 nominal predictors. Both datasets consist of training and testing subsets. For each dataset, we evenly split the full training subset (30,162 tuples of *Adult* and 3,763 tuples of *Mushroom*) into two portions, and distribute them to Alice and Bob, respectively. For each dataset, we randomly select 500 tuples from the testing subset to test the classifier models.

Table 5.6: Experiment Parameters for PPNB

Operator	κ	γ	β	τ	w	z	λ
$\times$		15	12	284	20	-	40
/	950			28			
power	910						
bit-length	-	-	-	-	-	45	

Implementation. We implemented the solutions by Java. All experiments were run on a machine of Intel® Xeon® 2.3GHz CPU (16 cores) with 128.0GB RAM running on Windows Server 2012. We downloaded the threshold Paillier Cryptosystem [10], and configured it to 1024-bit. We use CMP of the fast garbled circuit [11] that is one of the most efficient implementations for secure comparison. Table 5.6 gives the values of the security parameters we use.

In PPNB, we consider two parties Alice and Bob. We assume that each party has a subset of training data with the format (x_1, x_2, ..., x_m, y), where x_i's are predictor variables and y is the response variable. There is a testing dataset with the same format but y value missing. We assume that Alice and Bob would like to predict these y values via PPNB. Given a testing tuple with known values of predictor variables, PPNB predicts its y values using Equation 5.25. PPNB is detailed in Section 5.3.2.

[8] http://archive.ics.uci.edu/ml/datasets/Adult
[9] http://archive.ics.uci.edu/ml/datasets/Mushroom
[10] http://www.utdallas.edu/∼mxk093120/paillier/
[11] https://github.com/yhuang912/FastGC

We evaluate the effectiveness of PPNB based on our DAG model by two metrics. The first is *accuracy*. Given a testing dataset T, the accuracy is defined by

$$\mathrm{ACC}(T) = \frac{1}{|T|}\sum_{i=1}^{|T|}\begin{cases} 1, & \text{if } r_i.\hat{y} = r_i.y \\ 0, & \text{otherwise,} \end{cases} \tag{5.28}$$

where r_i is the i-th record in T, and $r_i.y$ and $r_i.\hat{y}$ are the real and predicted response values of r_i, respectively. The second is *similarity*, which measures the difference of prediction between the private setting and the non-private setting. Given a testing dataset T, the similarity is defined by

$$\mathrm{SIM}(T) = \frac{1}{|T|}\sum_{i=1}^{|T|}\begin{cases} 1, & \text{if } r_i.\widehat{y^P} = r_i.\widehat{y^{NP}} \\ 0, & \text{otherwise,} \end{cases} \tag{5.29}$$

where $r_i.\widehat{y^P}$ is the prediction for i-th tuple by PPNB based on DAG, and $r_i.\widehat{y^{NP}}$ is that by Naïve Bayes in non-private setting.

Our secure operators proposed in Section 4.1 have security parameters. We assume that values (i.e., a_1, a_2, b_1, b_2) are in the range of $[2^{-15}, 2^{15}]$ (i.e., $\gamma = 15$ in Lemma 4.2). If they are beyond the range, we truncate them to their nearest bounds. The security parameters like κ are restricted by different conditions in different secure operators. We set them based on these conditions, and thus their values vary from one operator to another. Table 5.6 gives the parameters and their values used in PPNB.

The Evaluation.

We evaluate our PPNB on the *Adult* and *Mushroom* datasets, which are described at the beginning of this section. Again, we include a benchmark, the non-private Naive Bayes that accesses the datasets of Alice and Bob directly. Its experimental outputs on Adult and Mushroom are *Adult-NP* and *Mushroom-NP*, respectively. The outputs by our private solution are *Adult-P* and *Mushroom-P*.

We first study the efficiency by varying the number of predictors(i.e., 30%/60%/100% predictor variables). Figure 5.4(a) reports the results for model construction. When the number of predictors increases, the elapsed time grows as expected. For the Mushroom dataset with all the predictors, the time by the non-private solution is 0.0160 sec, while that by our private solution is 39.1308 min. For the Adult dataset with all the predictors,

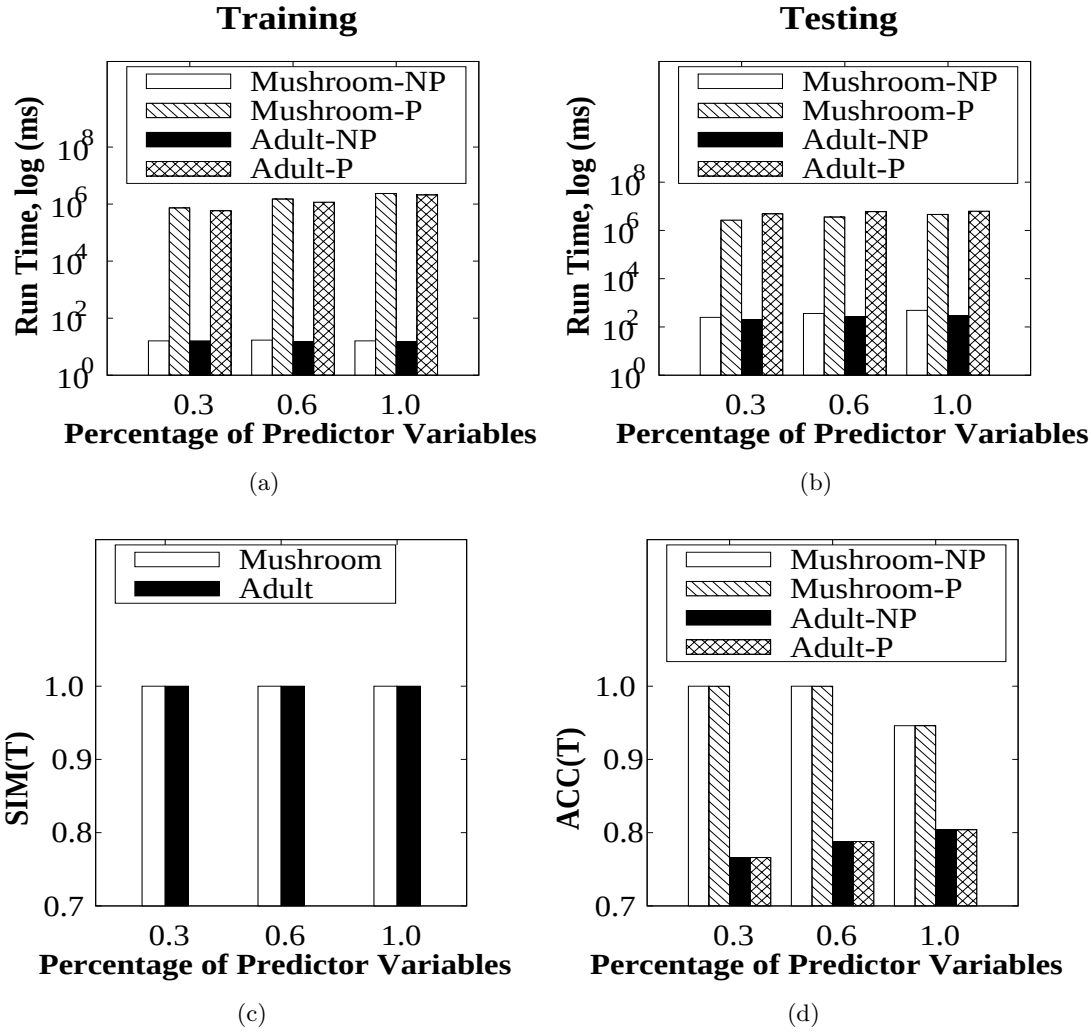


Figure 5.4: Naïve Bayes with/without privacy preservation

the time by the non-private solution is 0.0150 sec, while that by our private solution is 34.9021 min. Figure 5.4(b) is the experimental results of model evaluation. In the non-private setting, the prediction for one tuple on average takes 0.0008 sec for both datasets. In the private setting, the average time per tuple on Adult and Mushroom is 12.5039 sec and 9.1936 sec, respectively. The experimental results above show that our PPNB on the model construction and evaluation can be finished with reasonable time.

Next, we measure the similarity of the output between private solution and non-private solution. Figure 5.4(c) shows the results. When varying the number of predictors, the output of the two solutions is always the same. Therefore, the similarity is always 1. This verifies that the approximation errors of secure operators of our DAG model are low and negligible. Since the two solutions have the same output, their accuracy is also the same as in Figure 5.4(d). When the number of predictors increases, the accuracy on Adult dataset

increases accordingly, as expected. However, the accuracy on Mushroom dataset drops by 5.40% when the number of predictors increases from 60% to 100%. One possible reason is that some predictors are irrelevant (i.e., noise) to response values.

5.4 Chapter Summary

We propose three privacy-preserving classification algorithms to solve different classification problems with privacy preservation by applying our DAG model: privacy-preserving support vector machine (PPSVM), privacy-preserving kernel regression (PPKR), and privacy-preserving Naïve Bayes (PPNB). PPSVM uses arbitrarily partitioned data split between two parties to securely compute the Gram matrix. For the horizontal partitioned data split between two parties, PPKR can securely predict the response value and PPNB can securely build and evaluate the model. All proposed privacy-preserving classification algorithms are proven secure via simulation paradigm (Goldreich, 2004). Experiment results show that the privacy-preserving classification algorithms integrated with our DAG model are efficient in computation and also effective to protect data privacy. Moreover, the proposed privacy-preserving classification algorithms by DAG output the data mining results that are almost the same by non-privacy classification algorithms, without protecting data privacy.

Chapter 6

Privacy-Preserving Traveling Salesman Problem by DAG

We propose the DAG model which is the general model for privacy-preserving data mining in Chapter 4. The experiment results in Chapter 5 show that the privacy-preserving classification algorithms that can securely compute their tasks by applying secure operators of our DAG model. In this chapter, we show that our DAG model can also be the general model for privacy computation in wider applications. Evolutionary computation (EC) as a relatively new subfield of artificial intelligence is successfully applied to data analytics because of the robustness, fault tolerance, and scalability in computation (Martens et al., 2011; Freitas and Timmis, 2007; Freitas, 2002). Ant colony optimization (ACO) (Dorigo et al., 2006) is one of EC algorithms that can solve the traveling salesman problem (TSP). We apply our DAG model to solve the traveling salesman problem using ACO with privacy preservation. To the best of our knowledge, this is the first work to extend privacy protection to the EC algorithm.

In the following, we discuss how to integrate our DAG model into ACO with privacy preservation in Section 6.1. We summarize this chapter in Section 6.2.

6.1 Traveling Salesman Problem

The traveling salesman problem (TSP) requires that a salesman can find the shortest path, by which he visits each city only once and returns to the starting city. TSP is an NP-hard optimization problem. In the following, the ACO algorithm is discussed in Section 6.1.1. We propose privacy-preserving Traveling Salesman Problem (PPTSP) in applying ACO by DAG to find approximation solutions in Section 6.1.2. The security analysis and complexity analysis of PPTSP is given in Section 6.1.3. Lastly, we evaluate the performance of PPSTP in Section 6.1.4.

6.1.1 Algorithm

Ants randomly walk to locate food and to bring it back to their colony. The returning ant will drop pheromone on the walking trail that allows other ants to locate the food source and the colony location. Ant Colony Optimization (ACO) (Dorigo et al., 2006) is inspired by the behavior of biological ants – it creates artificial ants, which incrementally solve optimization problems in a way similar to biological ants finding the shortest path to the food source. Artificial ants choose solution trails also based on pheromone. The higher the pheromone of a decision point (i.e., a node visited previously by ants), the higher the probability of the point is likely chosen to visit. Once an artificial ant reaches the destination, the solution corresponding to the path visited by the ant is evaluated, and pheromones of the points along the path are increased. Pheromones of points, which are not visited, are evaporated. The ACO algorithm incrementally improves the solution. As the algorithm converges, the points of the optimal solution have the maximum pheromone, and all the other points have the minimum pheromone. ACO has been extensively applied to optimization problems, such as Traveling Salesman Problem (TSP) (Li, 2010), scheduling problem (Zhou et al., 2009), and vehicle routing problem (Mei et al., 2009).

ACO determines the next visiting point for an ant in a probabilistic way. Let x be the current position of an ant, and y be one of the next visiting point. Then, the probability of the ant visiting y is determined by two parameters: heuristic value (η_{xy}) where η_{xy} is a constant and pheromone value (τ_{xy}) is periodically updated using Eq. 6.2. Then, the probability is computed as follows.

$$\mathsf{p}_{\mathsf{xy}} = \frac{(\tau_{\mathsf{xy}}^{\alpha_\mathsf{c}})(\eta_{\mathsf{xy}}^{\beta_\mathsf{c}})}{\sum_{\mathsf{y} \in \mathsf{allowed}_\mathsf{y}} (\tau_{\mathsf{xy}}^{\alpha_\mathsf{c}})(\eta_{\mathsf{xy}}^{\beta_\mathsf{c}})}, \tag{6.1}$$

where α_c and β_c are control parameters to adjust the probability value and $\mathsf{allowed}_\mathsf{y}$ is the set of next visiting points. As ants reach the destinations, the pheromone is updated as follows:

$$\tau_{\mathsf{xy}} = (1 - \rho)\tau_{\mathsf{xy}} + \sum_{\mathsf{k}} \Delta_\mathsf{k}, \tag{6.2}$$

where ρ is a user-defined coefficient to control the pheromone evaporation rate,

$$\Delta_{\mathsf{k}} = \begin{cases} \mathsf{Q}/\mathsf{L}_{\mathsf{k}} & \text{if k-th ant visits trail xy in its tour} \\ 0 & \text{otherwise,} \end{cases} \tag{6.3}$$

Q is a constant value, and L_{k} is the total traversed distance of the k-th ant. Intuitively, in Equation 6.2, $(1-\rho)\tau_{\mathsf{xy}}$ evaporates the pheromone over time, and $\sum_k \Delta_k$ reinforce it if ants travel on the trail from point x to point y.

6.1.2 Privacy-Preserving Traveling Salesman Problem (PPTSP)

In privacy-preserving Traveling Salesman Problem (PPTSP), city locations that are private are distributed across Alice and Bob, and that Alice and Bob will not disclose the locations of the cities to each other. To keep the city locations private, we will formalize the ACO algorithm by our DAG model consisting of secure operators introduced in Chapter 4. Figure 6.1 (a) gives an example, in which filled blue circles represent cities of Alice and empty circles represent cities of Bob. ACO can be applied to find approximation solutions for TSP. Specifically, artificial ants can be created to visit the cities (e.g., Figure 6.1 (b) and Figure 6.1 (c)), and the shortest path for all the ants is taken as the solution output. Figure 6.2 gives the protocol for PPTSP. In PPTSP, we assume that all computations are done in the semi-honest model (Goldreich, 2004) which Alice and Bob strictly follow the protocol but they are curious about the private data of other party. We can use our DAG model to represent the protocol as follows.

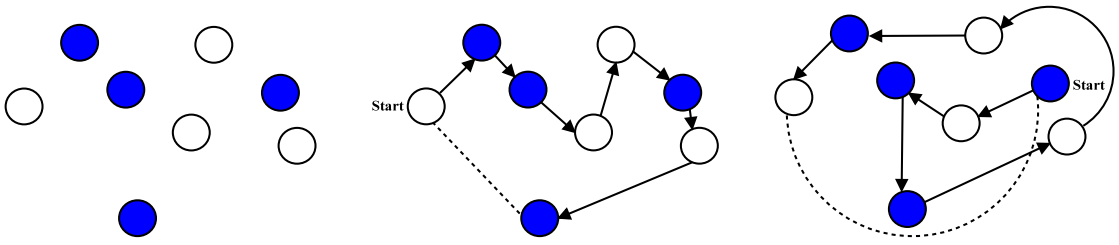


Figure 6.1: Blue filled circles represent cities of Alice and empty circles represent cities of Bob in Figure 6.1 (a). Examples of walking paths by ants formulated as an ACO problem in Figure 6.1 (b) and 6.1 (c)

Initialization. Alice and Bob configure the parameters of the protocol. They set α_c, β_c, ρ, and Q, which are needed to calculate the probability of traveling from one city to another (Equation 6.1). The pheromone τ_{xy} along the trail between city x and city y is initialized uniformly (e.g., equal to 1) and split into $\tau^{\mathsf{A}}_{\mathsf{xy}}$ and $\tau^{\mathsf{B}}_{\mathsf{xy}}$, such that $\tau_{\mathsf{xy}} = \tau^{\mathsf{A}}_{\mathsf{xy}} + \tau^{\mathsf{B}}_{\mathsf{xy}}$,

where τ_{xy}^{A} is distributed to Alice and τ_{xy}^{B} is distributed to Bob. The heuristic value η_{xy} is a constant. It is also split into η_{xy}^{A} and η_{xy}^{B}, and distributed to Alice and Bob, respectively.

Ant Walking. At each city, an ant needs to decide which is the next city to visit. The selection of the next city is a stochastic procedure. The probability of the next city is determined by Equation 6.1. Suppose that the ant is now at city x. The next city y to be visited is decided as follows. First, a value $p \in (0.0, 1.0)$ is selected based on Gaussian distribution. Without loss of generality, we assume that Alice selects p. Secondly, the probability p_{xy} of traveling from x to y is computed according to Equation 6.1. Clearly, the probability can be computed by a connection of operators of secure multiplication, secure addition, and secure division, of DAG model. Since every secure operator has two and only two output values, we assume that $p_{xy} = c_1 + c_2$, where c_1 and c_2 are the private outputs held by Alice and Bob, respectively. Then, Alice and Bob can privately determine whether $c_1 + c_2 \geq p$. That is, whether

$$c_1 - p \geq c_2. \tag{6.4}$$

For the privacy of c_1 and c_2, Alice and Bob apply CMP protocol (Section 3.1.10). If Inequality 6.4 holds, it shows that there is enough pheromone along the trail from x to y and the ant selects y as the next city to visit. Otherwise, another city is tested. Given a p value, if none of all the possible next cities can satisfy Inequality 6.4, Alice will select another p value.

Euclidean Distance. Alice and Bob can apply secure operators of DAG to securely compute Euclidean distance between two cities as follows. Given that an Euclidean function, $\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$, where Alice is in the city location (a_1, a_2) and Bob is in the city location (b_1, b_2). Alice and Bob first apply secure addition, secure minus, and secure multiplication operator to create two private distance portions t_1 and t_2 held by Alice and

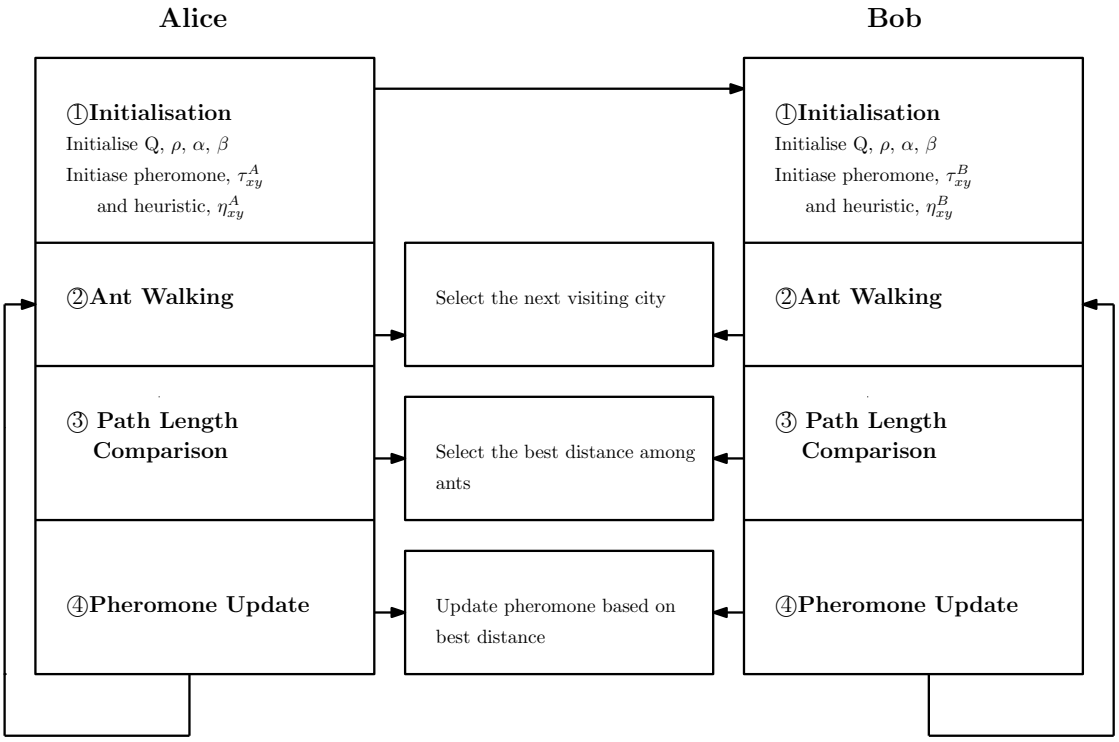


Figure 6.2: The SMC protocol for Traveling Salesman Problem (PPTSP)

Bob respectively, such that

$$\begin{aligned}(a_1 - b_1)^2 + (a_2 - b_2)^2 &= a_1^2 + a_2^2 - 2(a_1 \cdot b_1 + a_2 \cdot b_2) + b_1^2 + b_2^2 \\ &= a_1^2 + a_2^2 - 2(d_1 + d_2 + s_1 + s_2) + b_1^2 + b_2^2 \\ &= (a_1^2 + a_2^2 - 2(d_1 + s_1)) + (b_1^2 + b_2^2 - 2(d_2 + s_2)) \\ &= t_1 + t_2,\end{aligned}$$

where (d_1, d_2) are the outputs of secure multiplication on $a_1 \cdot b_1$, (s_1, s_2) are the outputs of secure multiplication on $a_2 \cdot b_2$, and (d_1, s_1) and (d_2, s_2) are held by Alice and Bob, respectively, They then apply secure power to get the distance as follows,

$$\sqrt{t_1 + t_2} = (t_1 + t_2)^{\frac{1}{2}} = c_1 + c_2,$$

where $c_1 + c_2$ is the distance between the two cities. Alice holds the distance c_1 and Bob holds the distance c_2.

Path Length Comparison. In PPTSP, each ant walks to find a solution. The best solution among all the ants will be selected. As discussed the Euclidean distance above,

Alice and Bob can use secure operators of DAG model to privately compute the distance of any two cities. Thus, we can compute the length of any path traveled by an ant by summing (i.e., by secure addition) the distances between each pair of neighboring cities along the path. Let P_1 be one path. Since the secure addition has two output values, we assume that the length of P_1 is equal to $c_{11}+c_{12}$. Let P_2 be another path, suppose that its length is equal to $c_{21}+c_{22}$. We can apply the CMP Protocol (Section 3.1.10) to privately compare whether P_1 is shorter than P_2 by checking

$$c_{11} - c_{21} \leq c_{22} - c_{12}. \tag{6.5}$$

If the above inequality holds, then P_1 is shorter, otherwise, longer. Based on the length comparison, we can then privately learn the shortest path.

Pheromone Update. The pheromone is updated between Alice and Bob based on Equation 6.2. The two parties first compute $\sum_k \Delta_{\mathsf{k}}$ (i.e., by secure addition and secure division) as shown in Equations 6.2 and 6.3. Since every secure operator always has two output values, we assume that $\sum_k \Delta_{\mathsf{k}} = \Delta^A + \Delta^B$, where Δ^A and Δ^B are privately held by Alice and Bob, respectively. Then, pheromone update is as follows:

$$\begin{cases} \tau_{\mathsf{xy}}^{\mathsf{A}} &= (1-\rho)\tau_{\mathsf{xy}}^{\mathsf{A}} + \Delta^A, \\ \tau_{\mathsf{xy}}^{\mathsf{B}} &= (1-\rho)\tau_{\mathsf{xy}}^{\mathsf{B}} + \Delta^B. \end{cases} \tag{6.6}$$

The correctness of the update can be easily verified, since

$$\begin{aligned} \tau_{xy} &= (1-\rho)\tau_{xy} + \sum_k \Delta_{\mathsf{k}} = (1-\rho)(\tau_{\mathsf{xy}}^{\mathsf{A}} + \tau_{\mathsf{xy}}^{\mathsf{B}}) + \sum_k \Delta_{\mathsf{k}} \\ &= \tau_{\mathsf{xy}}^{\mathsf{A}} + \tau_{\mathsf{xy}}^{\mathsf{B}}. \end{aligned}$$

In the above, the first equation holds, since it is the defined pheromone update procedure (Equation 6.2), the second equation holds, because at the initial stage $\tau_{xy} = \tau_{\mathsf{xy}}^{\mathsf{A}} + \tau_{\mathsf{xy}}^{\mathsf{B}}$, and the last equation holds because the pheromone update of Alice and Bob is specified in Equation 6.6.

The Convergence. Ants optimize the traveling distance for TSP incrementally. After the update of pheromone, all the ants will walk the cities again and find a new solution. Given the new solution and the previous solution, if the path length of the new solution

is higher than the previous one, then the ants stop and the previous solution is the final output.

Thus, given a set of cities that contain city locations split between Alice and Bob, privacy-preserving Traveling Salesman Problem (PPTSP) allows artificial ants to securely find the best traversed distance in which each city is visited. At the end of PPTSP, Alice learns nothing about city locations of Bob. Bob also learns nothing about city locations of Alice. PPTSP is detailed in Algorithm 7.

Algorithm 7: Privacy-Preserving Traveling Salesman Problem (PPTSP)

Input: Let G be the number of cities. The cities are split between Alice and Bob. Each city location are sensitive. Let $(ant_1, ant_2, \cdots, ant_m)$ be the list of m ants in ACO.

Output: The best traversed distance in G is $D_{opt} = c_1 + c_2$ where Alice holds the distance c_1 and Bob holds the distance c_2.

1 Alice and Bob each initialize α_c, β_c, ρ, and Q.
2 Initialize the number of iterations, w.
3 Set $j = 1$;
4 Set the best traversed distance, $D_{best} = 0$;
5 **repeat**
6 Set the current best traversed distance, $D_{curr} = 0$;
7 **for** *i = 1 to m* **do**
8 ant_i walks randomly among G cities split between Alice and Bob. ant_i always selects the next visited city with higher probability (Equation 6.1). The distance between two cities is calculated based on the Euclidean distance (refer to the distance calculation in a secure manner). ant_i can apply secure multiplication, secure addition, and secure division, of our DAG model to compute the probability.
9 ant_i can use CMP (Section 3.1.10) for probability comparison.
10 The total traversed distance by ant_i is $d_i^A + d_i^B$, held by Alice and Bob respectively.
11 {Note: Again, ant_i can use CMP (Section 3.1.10) for the distance comparison.}
12 **if** $d_i^A + d_i^B > D_{curr}$ **then**
13 Update the current best traversed distance, $D_{curr} = d_i^A + d_i^B$, by ant_i.
14 Set $ant_{best} = ant_i$;
15 **end**
16 **end**
17 **if** $D_{best} = 0$ *or* $D_{best} > D_{curr}$ **then**
18 $D_{best} = D_{curr}$;
19 **end**
20 Update the pheromone on the traversed trail using D_{best} by ant_{best} (Equation 6.2). ant_{best} can apply secure addition and division to update the pheromone as shown in Equations 6.2 and 6.3.
21 $j = j + 1$;
22 **until** $j > w$ *or* $D_{opt} < D_{curr}$;
23 $D_{best} = d_{best}^A + d_{best}^B = c_1 + c_2$, held by Alice and Bob respectively, where d_{best}^A is the best traversed distance of Alice and d_{best}^B is the best traversed distance of Bob.

6.1.3 Security Analysis and Complexity Analysis

We use time complexity and communication complexity to measure the performance of privacy-preserving Traveling Salesman Problem (PPTSP). PPTSP is a stochastic algorithm. For a simplicity, we measure time complexity and communication complexity for m ants in visiting G cities in one iteration.

Time Complexity. We measure the time complexity of PPTSP by modular exponentiations, since they consume most of the time. Alice and Bob require $\frac{Gm(G+1)}{2}$ times of secure power and that of secure multiplication on distance computation. They apply $\frac{Gm(G+1)}{2}$ times of secure division for probability calculation and that of CMP (Section 3.1.10) for probability comparison. Lastly, they require m times of CMP to get best traversed distance. The initialization of CMP (Ishai et al., 2003; Naor and Pinkas, 2001) takes some modular exponentiations. However, the initialization can be done before the protocol, and its cost can be amortized over all the runnings of PPTSP. Thus, we do not count its cost in PPTSP. Therefore, the number of modular exponentiations needed by PPTSP with m ants in visiting G cities in one iteration is $\frac{Gm(G+1)}{2} \cdot (9\omega + 99 + 72z)$, where ω is the number of iterations in secure division and secure power, and z is the maximum bit-length of input data.

Communication Complexity. We measure the communication complexity by the number of message bits passing between Alice and Bob. To compute the Euclidean distance, Alice and Bob need to transfer $\frac{Gm(G+1)}{2} \cdot (t_2(24z+28+8w)+6(\lambda+z+2)t_1 z)$ bits where t_1 is a security parameter (Note: t_1 is suggested to be 80 in practice (Kolesnikov et al., 2009)), t_2 is the message length in Paillier cryptosystem, ω is the number of iterations in secure division and secure power, z is the maximum bit-length of input data, and λ is the threshold (e.g., $\lambda = 40$). Alice and Bob transfer $\frac{Gm(G+1)}{2} \cdot (t_2(24z + 24 + 8w) + 9(\lambda + z + 2)t_1 z)$ bits in probability calculation and comparison. Lastly, they transfer $\frac{Gm(G+1)}{2} \cdot (3(\lambda + z + 2)t_1 z)$ bits for the distance comparison. The CMP initialization also has some communication cost. We do not involve it, since it can be done before running PPTSP. Therefore, the communication complexity of PPTSP with m ants in visiting G cities in one iteration is $\frac{Gm(G+1)}{2} \cdot (t_2(48z + 48 + 16w) + 18(\lambda + z + 2)t_1 z)$ bits.

Our proposed PPTSP is proven secure via simulation paradigm (refer to Section 2.6.1 for more details) in the following.

Theorem 6.1 *The PPTSP protocol is simulatable.*

Proof 6.1 *In PPTSP, given a set of cities, Alice and Bob use combined operators of* DAG *model to allow ants to securely find the best traversed distance in which each city is visited. Thus, the view of Alice and that of Bob can be simulated using a similar proof in Theorem 4.8. We omit the details here.*

6.1.4 Experiment and Discussion

In this section, we evaluate the performance of our proposed privacy-preserving Traveling Salesman Problem (PPTSP).

Dataset. We use three datasets [1] for PPTSP. The first is *Berlin52* that contains 52 cities. The second is *Eil* that contains 101 cities. The last is *A280* that contains 280 cities. Each dataset is split into two portions, and distribute them to Alice and Bob, respectively.

Table 6.1: Experiment Parameters for PPTSP

Operator	κ	γ	w	z	λ
$\times$	-	44	-	-	40
/	950		20	-	
power	910				
bit-length	-	-	-	45	

Implementation. We implemented the solutions by Java. All experiments were run on a machine of Intel® i7 2.7GHz CPU (2 cores) with 4.0GB RAM running on Windows XP. We downloaded the threshold Paillier Cryptosystem [2], and configured it to 1024-bit. We use CMP of the fast garbled circuit [3] that is one of the most efficient implementations for secure comparison. Table 6.1 gives the values of the security parameters we use. In the experiment, value initialization in PPTSP is in Table 6.2.

Table 6.2: Value Initialization in PPTSP

Parameter	Initialization value
pheromone, τ_{xy}	0.8
α_c	-0.2
β_c	9.6
Persistent value, ρ	0.3
Q	0.00001

[1] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/
[2] http://www.utdallas.edu/∼mxk093120/paillier/
[3] https://github.com/yhuang912/FastGC

We consider cities split between Alice and Bob. We assume that each party contains the locations of the cities which are confidential to each other. In PPTSP, ants allow to walk among the cities by finding the best traversed distance by which each city is visited. We include a benchmark, NPTSP, which is also an ACO-based solution for Traveling Salesman Problem without privacy protection. The parameters of NPTSP are given the same setting as PPTSP in Table 6.2. We compare PPTSP and NPTSP with respect to both effectiveness (i.e., the length of the path traveled by the salesman) and efficiency (i.e., the elapsed time for computing the traversed path). We use the following two metrics to evaluate the effectiveness of PPTSP and NPTSP.

$$\mathsf{Deviation}: \ \sigma_{\mathsf{aco}} = \sqrt{\frac{1}{\mathsf{N}}\sum_{\mathsf{i}=1}^{\mathsf{N}}(\mathsf{D}_{\mathsf{opt}} - \mathsf{D}_{\mathsf{i}})^2}, \tag{6.7}$$

$$\mathsf{Accuracy}: \ \gamma_{\mathsf{aco}} = 1 - \frac{\sigma_{\mathsf{aco}}}{\mathsf{D}_{\mathsf{opt}}}, \tag{6.8}$$

where $\mathsf{D}_{\mathsf{opt}}$ is the length of the suggested optimal path [4], N is the number of ants allowed in the experiments, and D_{i} is the length of the path computed by the i-th ant, for $i = 1, 2, \ldots, \mathsf{N}$.

Table 6.3: Performance metrics of PPTSP and NPTSP of 10 walking ants (N=10)

Dataset	$\mathsf{D}_{\mathsf{opt}}$	Algorithm	Avg D_{i}	Avg time (ms)	σ_{aco} / $\mathsf{D}_{\mathsf{opt}}$ (%)	γ_{aco} (%)
Berlin52	7542	PPTSP	8240.50	3,486,443	9.26	90.74
		NPTSP	8235.00	1249	9.19	90.81
Eil101	629	PPTSP	795.50	14,152,211	26.39	73.61
		NPTSP	795.00	3505	26.39	73.61
A280	2579	PPTSP	3178.12	128,652,372	23.23	76.77
		NPTSP	3170.00	26347	22.92	77.08
Berlin52 - 52 cities, Eil101 - 101 cities, and A280 - 280 cities $\mathsf{D}_{\mathsf{opt}}$ is the length of the suggested optimal path [4].						

Evaluation. To handle the randomness by probabilistically selecting the next visiting city by an ant, we run each experiment 10 times and report the average. We first fix the number of ants to 10, and compare PPTSP and NPTSP on the three datasets. Table 6.3 shows the results. Clearly, PPTSP and NPTSP are equally effective in terms of Deviation and Accuracy as defined in Equations 6.7 and 6.8, respectively. As for the efficiency, NPTSP is 3 orders of magnitude faster than PPTSP. This is reasonable, since PPTSP

[4] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/STSP.html

Table 6.4: Performance metrics of PPTSP with the number of ants

Dataset	D_{opt}	N (Ant No.)	Avg D_i	Avg time (ms)	σ_{aco} / D_{opt} (%)	γ_{aco} (%)
Berlin52	7542	1	9337.66	510,727	23.80	76.20
		2	8474.60	690,900	12.37	87.63
		5	8258.14	1,906,668	9.50	90.50
		10	8240.50	3,486,443	9.26	90.74
Eil101	629	1	831.77	1,905,322	32.24	67.76
		2	796.28	2,533,562	26.60	73.40
		5	803.99	7,015,963	27.80	72.20
		10	795.50	14,152,211	26.39	73.61
A280	2579	1	3489.76	28,408,946	35.31	64.69
		2	3292.39	26,094,679	27.66	72.34
		5	3178.14	79,345,737	23.23	76.77
		10	3178.12	128,652,372	23.23	76.77
Berlin52 - 52 cities, Eil101 - 101 cities, and A280 - 280 cities D_{opt} is the length of the suggested optimal path [4].						

needs expensive cryptographic modular exponentiations, which are necessary to ensure the data privacy of Alice and Bob.

Next, we vary the number of ants from 1 to 10. Table 6.4 reports the results. As the number of ants increases, the elapsed time of PPTSP grows. This is as expected, since each ant needs time to find its solution. When the number of ants increases, the number of traveled paths also increases. Therefore, the quality (by deviation and accuracy) of the best path output by PPTSP also improves.

6.2 Chapter Summary

We propose privacy-preserving traveling salesman problem (PPTSP) using ant colony optimization (ACO) by DAG to solve the traveling salesman problem with privacy preservation. In the problem, the cities contain locations which are private (i.e., sensitive). The city locations are split between Alice and Bob. At the end of the execution of PPTSP, the city locations of Alice are confidential to Bob. Bob also learns nothing about the city locations of Alice. The PPTSP protocol is proven secure via simulation paradigm (Goldreich, 2004). The experiment results show that the PPTSP solution outputs the traveling path with a length very close to the optimal solution and the DAG model is effective in protecting data privacy.

Chapter 7

Conclusions

In a modern information-driven society, data from individuals and companies may contain various kinds of private information. Rapid advances in automated data collection tools and data storage technology has led to the wide availability of a huge amount of the distributed data owned by different parties. Data mining algorithms can be used to mine the distributed data that can lead to new insights and economic advantages that usually cannot be mined from the data of any single party; e.g., two hospitals can jointly discover more useful and meaningful knowledge, rules or patterns from their joint data of the patient records, which they would not get otherwise. However, data sharing poses privacy concerns such as privacy violation and data protection laws on private and sensitive data. Therefore many parties have strong willingness to share their data to data mining algorithms that can protect data privacy to get a global view of data for knowledge discovery. This motivates many to propose privacy-preserving data mining (PPDM) algorithms to address the issue (Lindell and Pinkas, 2000; Agrawal and Srikant, 2000). Many PPDM algorithms apply the randomization or secure multi-party computation (SMC) to protect data privacy of the participating parties during the mining process.

In recent years, PPDM has become more important by which the increase of sophisticated data mining algorithms can leverage bigger data that contain more private information. Many conventional data mining algorithms have been enhanced so as to provide privacy preservation. These algorithms include such as support vector machine (SVM), Naïve Bayes, k-means clustering, association rule mining, decision trees and many more. These algorithms can mine the distributed data split in a specific partition, either in vertical or horizontal partition, and both of them. Each partition is held by a participating party. Many secure building blocks of SMC based on the semi-honest model have been proposed in the algorithms.

However, many SMC solutions are ad-hoc and specific to tasks. They cannot be directly applied to other tasks. Another limitation of the SMC solutions is provided by the limited set of secure operators to support more functions in data mining primitives. In our work, we propose DAG which is the general model for privacy-preserving data mining to address the limitations in Chapter 4. Our DAG model consists of a set of secure operators. We use the hybrid model that combines the semi-homomorphic encryption protocol and the circuit approach to propose the secure operators. The secure operators can be pipelined together to compute many functions in data mining primitives. The theoretical and empirical proofs of the secure operators are discussed in detail.

As case studies in data mining, we first apply our DAG model into three different classification algorithms, support vector machine (SVM), kernel regression, and Naïve Bayes, to solve different classification problems with privacy preservation in Chapter 5. Therefore, our DAG is a general model yet efficient for privacy-preserving data mining. For a case study in an other application domain, we also apply the DAG model into ant colony optimization (ACO) to solve the traveling salesman problem (TSP) without disclosing any city location in Chapter 6. This shows that our DAG model can support wider applications in protecting data privacy.

7.1 Summary of Research Contributions

We summarize our contributions in detail as follows:

(i) We have proposed DAG – the general model for privacy-preserving data mining. Our DAG model consists of a set of secure operators in Chapter 5. Currently DAG supports 9 operators that include secure addition, secure minus, secure multiplication, secure division, secure log, secure power, secure bit-length, secure max, and secure max location. Each operator protocol is strictly proven secure via simulation paradigm (Goldreich, 2004). We also use the other two security measurements, statistical indistinguishability(Agrawal et al., 2003) and computational indistinguishability (Goldreich, 2004), to measure security of our secure operators. Secure operators of DAG can provide a complete privacy under the semi-honest model.

(ii) Our secure operators can support non-integer and integer values. To support non-integer values, we have provided various techniques to convert non-integer values

into integer values with the conversion errors bounded to an acceptable threshold. Each secure operator can perform a function as specified. The error analysis and complexity analysis (e.g., the error bounds and the time complexity) of each secure operators of DAG have been discussed in detail. Moreover, secure operators can also be pipelined together to perform various functions in data mining tasks. We have analyzed the error bounds of the connection operators. The connection of the secure operators has also been proven secure via simulation paradigm.

(iii) In the DAG model, we have applied the hybrid SMC model that combines the semi-homomorphic encryption protocol and the circuit approach for our secure operators. For the circuit approach, the secure operators only use CMP (integer comparison circuit) to compare values. Lastly, we have evaluated the performance of secure operators of DAG. The experiment results have showed that our secure operators are efficient in computation and effective in protecting data privacy. Thus, our secure operators have been proven secure and efficient via the theoretical and experimental proofs. Our DAG model can also support any new operator that is derived based on the definition of secure operator (Definition 4.1).

(iv) We have proposed privacy-preserving classification algorithms by DAG in Chapter 5. The first is privacy-preserving support vector machine (PPSVM). PPSVM has securely computed the gram matrix from arbitrarily partitioned data split between two parties. We have further extended PPSVM to support on multiple parties. The second is privacy-preserving kernel regression (PPKR) that has securely predicted response values based on horizontally partitioned data split between two parties. When training data size is 689,093, one predictor in non-private setting of kernel regression takes 5.93 sec, and that by PPKR takes 12.38 sec. The last algorithm, privacy-preserving Naïve Bayes (PPNB) has built a secure model from horizontal partitioned data split between two parties. PPNB has also securely predicted response values based on the secure model. The time complexity and communication complexity of each privacy-preserving classification algorithm have been discussed in detail. All privacy-preserving algorithms have been proven secure via simulation paradigm. The experiment results have showed that our DAG model can be the general model yet efficient for privacy-preserving data mining.

(v) We have proposed privacy-preserving traveling salesman problem (PPTSP) by DAG in Chapter 6. In the traveling salesman problem (TSP), PPTSP has found the approximate optimal traveled distance in visiting all a given set of cities by applying ant colony optimization (ACO). In the end of the execution of PPTSP, Alice learns nothing about the city locations of Bob. The city locations of Bob are also confidential from Alice. PPTSP has been proven secure via simulation paradigm. The time complexity and communication complexity of PPTSP have been discussed in detail. The experiment results have showed that PPTSP has the similar approximate optimal distance traversed by a salesman in ACO, without privacy preservation. Thus, our DAG model apart from the data mining tasks can also support other application areas. The experiments in various application domains have showed that our DAG model can serve as the general model for privacy computation in wider applications.

7.2 Future Work

In this thesis, we have discussed a variety of privacy issues and open problems remaining in privacy-preserving data mining (PPDM). We will continue to investigate our DAG model to address many issues raised in PPDM and to support other application domains. We briefly discuss our future work as follows,

(i) We will continue to investigate new useful operators that can be added to the DAG model to support wider applications. We will also continue to integrate DAG into different data mining algorithms and other domain applications.

(ii) Our secure operators of DAG use the hybrid SMC model that combines the homomorphic encryption protocol and the circuit approach. In SMC, modulus exponentiations consume most of the computation time. To reduce the computation cost, we will investigate to enhance the cryptographic primitives of the homomorphic encryption scheme that can run more efficiently in computation. Currently our DAG model supports threshold Paillier cryptosystem. To make the model more robust, we will plan to integrate DAG with different semi-homomorphic encryption schemes. We will also interested in applying fully-homomorphic schemes into our DAG model.

(iii) Many secure building blocks of the privacy-preserving data mining algorithms randomize or encrypt whole data during the mining process. Our DAG model is also

working in such a similar way as other secure building blocks. In some situations, data may contain some private and non-private information. This raises interesting questions to be answered: Is any possibility that a secure building block only randomize or encrypt the private data in PPDM? Can computation cost and communication cost be reduced as the secure building block only protects the private data? We will investigate to enhance our DAG model to support both private and non-private data during the mining process.

(iv) Our DAG model has showed that it can support wider applications. Thus, we need to consider the malicious model that DAG can withstand the malicious attacks in some application domains. We will propose a DAG model under the malicious model. Another improvement is that we will extend the two-party protocol of secure operators of DAG to the multi-party protocol. Moreover, we will also investigate to parallelize the multi-party protocol of secure operators.

(v) Graph is a general data structure that exists in various domains, e.g., pattern recognition, computational biology, computer vision, web analysis and text retrieval. One of the graph examples is protein-protein interaction (PPI) network. Two medical research institutions each have a PIP network that contains the private deoxyribonucleic acid (DNA) information. They would be interested to discover a new drug of the genetic disease via interaction of their PIP networks with privacy preservation on DNA. Therefore, we are interested to apply our DAG model into graph for secure computation. In addition, we will investigate the possibility to combine the DAG model and the privacy-preserving data publishing (PPDP) techniques to protect data privacy in graph. In PPDP, an individual can apply some privacy preservation techniques to modify his/her own private data before publishing them to data miners. The published data requires to remain useful for data mining. We will apply our DAG model and PPDP to address privacy issues in social network (e.g., Facebook and Twitter). We are also interested to integrate the DAG model into deep learning algorithms that can train social network data with privacy preservation.

(vi) We are interested to integrate our DAG model in other application domains that involve technologies such as peer-to-peer (P2P), mobile computing, cloud computing, and many more. For example, big data analytics with privacy concerns in cloud

computing is still a challenging and ongoing research topic. We will continue to enhance our DAG model for big data analytics. Besides the computation cost, the communication cost and the size of our DAG model are important especially in mobile computing with limited resources (e.g., small space in memory). Therefore, we will also propose a DAG model to adapt in such a restrained environment.

References

Abdalla, M., Bellare, M. and Rogaway, P. (1999). DHAES: an encryption scheme based on the diffie-hellman problem, *IACR Cryptology ePrint Archive* **1999**.

Adam, N. R. and Wortmann, J. C. (1989). Security-control methods for statistical databases: A comparative study, *ACM Comput. Surv.* pp. 515–556.

Aggarwal, C. C. and Yu, P. S. (2008). A general survey of privacy-preserving data mining models and algorithms, *Privacy-Preserving Data Mining - Models and Algorithms*, Vol. 34 of *Advances in Database Systems*, pp. 11–52.

Agrawal, D. and Aggarwal, C. C. (2001). On the design and quantification of privacy preserving data mining algorithms, *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, Santa Barbara, California, United States, pp. 247–255.

Agrawal, R., Evfimievski, A. V. and Srikant, R. (2003). Information sharing across private databases, *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA*, pp. 86–97.

Agrawal, R., Gehrke, J., Gunopulos, D. and Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications, *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, Seattle, Washington, United States, pp. 94–105.

Agrawal, R., Imielinski, T. and Swami, A. N. (1993). Mining association rules between sets of items in large databases, *in* P. Buneman and S. Jajodia (eds), *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, D.C., pp. 207–216.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases, *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 487–499.

Agrawal, R. and Srikant, R. (2000). Privacy-preserving data mining, *Proceedings of the ACM international Conference on Management of Data*, Dallas, Texas, United States, pp. 439–450.

Atallah, M. J., Bykova, M., Li, J., Frikken, K. B. and Topkara, M. (2004). Private collaborative forecasting and benchmarking, *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society, WPES 2004, Washington, DC, USA, October 28, 2004*, pp. 103–114.

Bar-Ilan, J. and Beaver, D. (1989). Non-cryptographic fault-tolerant computing in constant number of rounds of interaction, *Proceedings of the eighth annual ACM Symposium on Principles of distributed computing*, Edmonton, Alberta, Canada, pp. 201–209.

Barni, M., Orlandi, C. and Piva, A. (2006). A privacy-preserving protocol for neural-network-based computation, *Proceedings of the 8th ACM Workshop on Multimedia and Security*, Geneva, Switzerland, pp. 146–151.

Baudron, O., Fouque, P., Pointcheval, D., Stern, J. and Poupard, G. (2001). Practical multi-candidate election system, *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing, PODC 2001, Newport, Rhode Island, USA*, pp. 274–283.

Ben-David, A., Nisan, N. and Pinkas, B. (2008). Fairplaymp: a system for secure multi-party computation, *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, Alexandria, Virginia, USA, pp. 257–266.

Ben-Or, M., Goldwasser, S. and Wigderson, A. (1988). Completeness theorems for non-cryptographic fault-tolerant distributed computation, *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, Chicago, Illinois, United States, pp. 1–10.

Benaloh, J. C. (1987). Secret sharing homomorphisms: keeping shares of a secret secret, *Proceedings on Advances in cryptology—CRYPTO86*, Springer-Verlag, Santa Barbara, California, United States, pp. 251–260.

Berkhin, P. (2002). Survey of clustering data mining techniques, *Technical report*, Accrue Software.

Bertino, E., Fovino, I. N. and Provenza, L. P. (2005). A framework for evaluating privacy preserving data mining algorithms, *Data Min. Knowl. Discov.*, pp. 121–154.

Blum, M. and Goldwasser, S. (1984). An efficient probabilistic public-key encryption scheme which hides all partial information., *Proceedings of CRYPTO on Advances in cryptology*, pp. 289–302.

Bodrato, M. (2010). A strassen-like matrix multiplication suited for squaring and higher power computation, *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*, ACM, pp. 273–280.

Boneh, D. (1998). The decision diffie-hellman problem, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA*, pp. 48–63.

Boneh, D., Goh, E. and Nissim, K. (2005). Evaluating 2-dnf formulas on ciphertexts, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA*, pp. 325–341.

Boser, B. E., Guyon, I. M. and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers, *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, Pittsburgh, Pennsylvania, United States, pp. 144–152.

Brakerski, Z., Gentry, C. and Vaikuntanathan, V. (2012). (leveled) fully homomorphic encryption without bootstrapping, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA*, pp. 309–325.

Brandt, F. and Sandholm, T. (2008). On the existence of unconditionally privacy-preserving auction protocols, *ACM Trans. Inf. Syst. Secur.* **11**(2).

Brassard, G., Crépeau, C. and Robert, J. (1986). All-or-nothing disclosure of secrets, *Advances in Cryptology - Santa Barbara, California, USA*, pp. 234–238.

Breiman, L., Friedman, J., Stone, C. J. and Olshen, R. (1984). *Classification and Regression Trees*, Chapman & Hall/CRC, Belmont, CA: Wadsworth Int.

Brickell, J., Porter, D. E., Shmatikov, V. and Witchel, E. (2007). Privacy-preserving remote diagnostics, *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, ACM, New York, NY, USA, pp. 498–507.

Bunn, P. and Ostrovsky, R. (2007). Secure two-party k-means clustering, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, Alexandria, Virginia, USA*, pp. 486–497.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery* **2**(2): 121–167.

Chaum, D., Crépeau, C. and Damgård, I. (1988). Multiparty unconditionally secure protocols (extended abstract), *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, Chicago, Illinois, USA*, pp. 11–19.

Chor, B. and Kushilevitz, E. (1993). A communication-privacy tradeoff for modular addition, *Information Processing Letters* **45**(4): 205–210.

Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X. and Zhu, M. Y. (2002). Tools for privacy preserving distributed data mining, *In SIGKDD Explorations* pp. 28–34.

Congress, U. S. (1996). Health insurance portability and accountability act of 1996, *U.S. Congress* .

Coppersmith, D. and Winograd, S. (1987). Matrix multiplication via arithmetic progressions, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, USA*, pp. 1–6.

Coron, J., Lepoint, T. and Tibouchi, M. (2013). Batch fully homomorphic encryption over the integers, *IACR Cryptology ePrint Archive* p. 36.

Coron, J., Naccache, D. and Tibouchi, M. (2012). Public key compression and modulus switching for fully homomorphic encryption over the integers, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK*, pp. 446–464.

Cramer, R. and Damgård, I. (2001). Secure distributed linear algebra in a constant number of rounds, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA*, pp. 119–136.

Cramer, R., Damgård, I. and Nielsen, J. B. (2001). Multiparty computation from threshold homomorphic encryption, *Advances in Cryptology - EUROCRYPT 2001, International*

Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, pp. 280–299.

Cramer, R. and Shoup, V. (1998). A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA*, pp. 13–25.

Dahl, M., Ning, C. and Toft, T. (2012). On secure two-party integer division, *Financial Cryptography and Data Security - 16th International Conference, FC 2012, Kralendijk, Bonaire*, pp. 164–178.

Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J. B. and Toft, T. (2006). Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation, *Third Theory of Cryptography Conference, TCC 2006, New York, USA*, pp. 285–304.

Damgård, I. and Jurik, M. (2001). A generalisation, a simplification and some applications of paillier's probabilistic public-key system, *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea*, pp. 119–136.

Das, A., Ng, W.-K. and Woon, Y.-K. (2001). Rapid association rule mining, *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, ACM Press, Atlanta, Georgia, USA, pp. 474–481.

Domingos, P. M. and Pazzani, M. J. (1996). Beyond independence: Conditions for the optimality of the simple bayesian classifier, *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy*, pp. 105–112.

Dorigo, M., Birattari, M. and Stutzle, T. (2006). Ant colony optimization – artificial ants as a computational intelligence technique, *IEEE Computational Intelligence Magazine* **1**: 28–39.

Du, W. and Atallah, M. (2001a). Secure multi-party computation: A review and open problems, *Technical report*, CERIAS Tech. Report.

Du, W. and Atallah, M. J. (2001b). Privacy-preserving cooperative statistical analysis, *17th Annual Computer Security Applications Conference (ACSAC 2001), New Orleans, Louisiana, USA*, pp. 102–110.

Du, W., Han, Y. and Chen, S. (2004). Privacy-preserving multivariate statistical analysis: Linear regression and classification, *Proceedings of the 4th SIAM International Conference on Data Mining*, Lake Buena Vista, Florida, pp. 222–233.

Du, W. and Zhan, Z. (2002). Building decision tree classifier on private data, *Proceedings of the IEEE International Conference on Privacy, Security and Data Mining*, Maebashi City, Japan, pp. 1–8.

Du, W. and Zhan, Z. (2003). Using randomized response techniques for privacy-preserving data mining, *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, Washington, D.C., USA, pp. 505–510.

ECHR (2014). Handbook on european data protection law.
URL: *http://www.echr.coe.int/Documents/Handbook_data_protection_ENG.pdf*

Emekçi, F., Sahin, O. D., Agrawal, D. and El Abbadi, A. (2007). Privacy preserving decision tree learning over multiple parties, *Data Knowl. Eng.* **63**(2): 348–361.

Ester, M., Kriegel, H.-P., Sander, J. and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise, *Proceedings of 2nd Int. Conf. on Knowledge Discovery and Data Mining*, Menlo Park, CA, pp. 226–231.

Even, S., Goldreich, O. and Lempel, A. (1985). A randomized protocol for signing contracts, *Communications of the ACM* **28**(6): 637–647.

Evfimievski, A., Srikant, R., Agrawal, R. and Gehrke, J. (2002). Privacy preserving mining of association rules, *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, Edmonton, Alberta, Canada, pp. 217–228.

Fouque, P., Poupard, G. and Stern, J. (2000). Sharing decryption in the context of voting or lotteries, *Financial Cryptography, 4th International Conference, FC.*, pp. 90–104.

Frankel, Y., MacKenzie, P. D. and Yung, M. (1998). Robust efficient distributed rsa-key generation, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA*, pp. 663–672.

Freeman, D. M. (2010). Converting pairing-based cryptosystems from composite-order groups to prime-order groups, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera*, pp. 44–61.

Freitas, A. A. (2002). A survey of evolutionary algorithms for data mining and knowledge discovery, *Advances in Evolutionary Computation*, Springer-Verlag, pp. 819–845.

Freitas, A. A. and Timmis, J. (2007). Revisiting the foundations of artificial immune systems for data mining, *IEEE Transactions on Evolutionary Computation* **11**(4): 521–540.

Friedman, J., Hastie, T. and Tibshirani, R. (2001). *The elements of statistical learning*, Springer series in statistics.

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

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA*, pp. 169–178.

Gentry, C. and Halevi, S. (2011). Implementing gentry's fully-homomorphic encryption scheme, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia*, pp. 129–148.

Gjøsteen, K. (2005). Homomorphic cryptosystems based on subgroup membership problems, *Progress in Cryptology - Mycrypt 2005, First International Conference on Cryptology in Malaysia, Kuala Lumpur, Malaysia*, pp. 314–327.

Goethals, B., Laur, S., Lipmaa, H. and Mielikäinen, T. (2004). On private scalar product computation for privacy-preserving data mining, *Information Security and Cryptology - ICISC 2004, 7th International Conference, Seoul, Korea*, pp. 104–120.

Goil, S., Nagesh, H. and Choudhary, A. (1999). Mafia: Efficient and scalable subspace clustering for very large data sets, *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 443–452.

Goldreich, O. (2001). *The Foundations of Cryptography - Volume 1, Basic Techniques*, Cambridge University Press.

Goldreich, O. (2004). *The Foundations of Cryptography - Volume 2, Basic Applications*, Cambridge University Press.

Goldwasser, S. and Micali, S. (1982). Probabilistic encryption and how to play mental poker keeping secret all partial information, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, San Francisco, California, USA*, pp. 365–377.

Goldwasser, S., Micali, S. and Rackoff, C. (1989). The knowledge complexity of interactive proof systems, *SIAM Journal on Computing* **18**(1): 186–208.

Guha, S., Rastogi, R. and Shim, K. (1998). Cure: an efficient clustering algorithm for large databases, *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, Seattle, Washington, United States, pp. 73–84.

Han, J., Kamber, M. and Pei, J. (2006). *Data mining: Concepts and techniques*, Morgan kaufmann.

Han, J., Pei, J. and Yin, Y. (2000). Mining frequent patterns without candidate generation, *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, ACM Press, Dallas, Texas, United States, pp. 1–12.

Han, S. and Ng, W. K. (2008). Preemptive measures against malicious party in privacy-preserving data mining, *Proceedings of the SIAM International Conference on Data Mining, SDM 2008, Atlanta, Georgia, USA*, pp. 375–386.

Han, S., Ng, W. K., Wan, L. and Lee, V. C. S. (2010). Privacy-preserving gradient-descent methods, *IEEE Trans. Knowl. Data Eng.* pp. 884–899.

Han, S., Ng, W. K. and Yu, P. S. (2009). Privacy-preserving singular value decomposition, *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, Shanghai, China*, pp. 1267–1270.

Hesse, W., Allender, E. and Barrington, D. A. M. (2002). Uniform constant-depth threshold circuits for division and iterated multiplication, *J. Comput. Syst. Sci.* pp. 695–716.

Hinneburg, A. and Keim, D. A. (1998). An efficient approach to clustering in large multimedia databases with noise, *Knowledge Discovery and Data Mining*, New York, USA, pp. 58–65.

Hipp, J., Guntzer, U. and Nakhaeizadeh, G. (2000). Algorithms for association rule mining - a general survey and comparison, *SIGKDD Explorations Newsletter* **2**(1): 58–64.

Hirt, M. and Sako, K. (2000). Efficient receipt-free voting based on homomorphic encryption, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium*, pp. 539–556.

Howgrave-Graham, N. (2001). Approximate integer common divisors, *Cryptography and Lattices, International Conference, CaLC 2001, Providence, RI, USA*, pp. 51–66.

Hsu, H.-H. (2006). *Advanced data mining technologies in bioinformatics*, Idea Group,U.S.

Huang, Y., Evans, D., Katz, J. and Malka, L. (2011). Faster secure two-party computation using garbled circuits, *USENIX Security Symposium*.

Huang, Y., Malka, L., Evans, D. and Katz, J. (2011). Efficient privacy-preserving biometric identification, *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA*.

Huang, Z., Du, W. and Chen, B. (2005). Deriving private information from randomized data, *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, Baltimore, Maryland, pp. 37–48.

Ioannidis, I., Grama, A. and Atallah, M. (2002). A secure protocol for computing dot-products in clustered and distributed environments, *Proceedings of the International Conference on Parallel Processing*, Vancouver, British Columbia, Canada, pp. 379–384.

Ishai, Y., Kilian, J., Nissim, K. and Petrank, E. (2003). Extending oblivious transfers efficiently, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA*, pp. 145–161.

Ishai, Y. and Kushilevitz, E. (1997). Private simultaneous messages protocols with applications, *Theory of Computing and Systems, 1997., Proceedings of the Fifth Israeli Symposium*, pp. 174–183.

Jagannathan, G., Pillaipakkamnatt, K. and Wright, R. N. (2006). A new privacy-preserving distributed k-clustering algorithm, *Proceedings of the Sixth SIAM International Conference on Data Mining, Bethesda, MD, USA*, pp. 494–498.

Jagannathan, G. and Wright, R. N. (2005). Privacy-preserving distributed k-means clustering over arbitrarily partitioned data, *Proceedings of the 8th ACM International Conference on Knowledge Discovery in Data Mining*, Chicago, Illinois, USA, pp. 593–599.

Jha, S., Kruger, L. and McDaniel, P. (2005). Privacy preserving clustering, *Computer Security - ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy*, pp. 397–417.

Jiang, W. and Clifton, C. (2007). Ac-framework for privacy-preserving collaboration, *Proceedings of the Seventh SIAM International Conference on Data Mining, Minneapolis, Minnesota, USA*, pp. 47–56.

Kantarcioglu, M. and Clifton, C. (2003). Privacy preserving naive bayes classifier for horizontally partitioned data, *IEEE ICDM Workshop on Privacy Preserving Data Mining*, Melbourne, FL, pp. 3–9.

Kantarcioglu, M. and Clifton, C. (2004). Privacy-preserving distributed mining of association rules on horizontally partitioned data, *IEEE Trans. Knowl. Data Eng.* **16**(9): 1026–1037.

Kantarcioglu, M. and Kardes, O. (2006). Privacy-preserving data mining in malicious model, *Technical report*.

Kantarcioglu, M. and Kardes, O. (2007). Privacy-preserving data mining applications in the malicious model, *Workshops Proceedings of the 7th IEEE International Conference on Data Mining, Omaha, Nebraska, USA*, pp. 717–722.

Kargupta, H., Datta, S., Wang, Q. and Sivakumar, K. (2003). On the privacy preserving properties of random data perturbation techniques, *Proceedings of the Third IEEE International Conference on Data Mining*, Melbourne, Folrida, pp. 99–106.

Karypis, G., Han, E. and Kumar, V. (1999). Chameleon: Hierarchical clustering using dynamic modeling, *IEEE Computer* **32**(8): 68–75.

Kerschbaum, F. (2008). Practical privacy-preserving benchmarking, *Proceedings of The IFIP TC-11 23rd International Information Security Conference, IFIP 20th World Computer Congress, IFIP SEC 2008, Milano, Italy*, pp. 17–31.

Kiltz, E., Leander, G. and Malone-Lee, J. (2005). Secure computation of the mean and related statistics, *Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA*, pp. 283–302.

Kolesnikov, V., Sadeghi, A. and Schneider, T. (2009). Improved garbled circuit building blocks and applications to auctions and computing minima, *Cryptology and Network Security, 8th International Conference, CANS 2009, Kanazawa, Japan*, pp. 1–20.

Kolesnikov, V. and Schneider, T. (2008). Improved garbled circuit: Free XOR gates and applications, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland*, pp. 486–498.

Laur, S., Lipmaa, H. and Mielikainen, T. (2006). Cryptographically private support vector machines, *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, USA, pp. 618–624.

Li, Y. (2010). Solving tsp by an aco-and-boa-based hybrid algorithm, *International Conference on Computer Application and System Modeling (ICCASM)*, pp. 189–192.

Lin, X., Clifton, C. and Zhu, M. Y. (2005). Privacy-preserving clustering with distributed EM mixture modeling, *Knowl. Inf. Syst.* **8**(1): 68–81.

Lindell, Y. and Pinkas, B. (2000). Privacy preserving data mining, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA*, pp. 36–54.

Lindell, Y. and Pinkas, B. (2002). Privacy preserving data mining, *J. Cryptology* **15**(3): 177–206.

Lindell, Y. and Pinkas, B. (2007). An efficient protocol for secure two-party computation in the presence of malicious adversaries, *Advances in Cryptology - EUROCRYPT 2007,*

26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, pp. 52–78.

Lindell, Y. and Pinkas, B. (2008). Secure multiparty computation for privacy-preserving data mining, *IACR Cryptology ePrint Archive*.

Lindell, Y. and Pinkas, B. (2009). A proof of security of yao's protocol for two-party computation, *J. Cryptology* **22**(2): 161–188.

Machanavajjhala, A., Kifer, D., Gehrke, J. and Venkitasubramaniam, M. (2007). *L*-diversity: Privacy beyond *k*-anonymity, *ACM Transactions on Knowledge Discovery from Data (TKDD)* **1**(1).

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, pp. 281–297.

Malkhi, D., Nisan, N., Pinkas, B. and Sella, Y. (2004). Fairplay - secure two-party computation system, *Proceedings of the 13th USENIX Security Symposium, San Diego, CA, USA*, pp. 287–302.

Martens, D., Baesens, B. and Fawcett, T. (2011). Editorial survey: swarm intelligence for data mining, *Machine Learning* **82**(1): 1–42.

McSherry, F. and Talwar, K. (2007). Mechanism design via differential privacy, *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), Providence, RI, USA*, pp. 94–103.

Mei, D., Shi, X. and Zhao, F. (2009). An improved aco algorithm for vehicle scheduling problem in military material distribution, *IEEE International Conference on Grey Systems and Intelligent Services*, pp. 1596–1600.

Menezes, A., van Oorschot, P. C. and Vanstone, S. A. (1996). *Handbook of Applied Cryptography*, CRC Press.

Mitchell, T. M. (1997). *Machine Learning*, McGraw-Hill.

Moskowitz, L. and Chang, I. S. (2000). A decision theoretical based system for information downgrading, *Technical report*, DTIC Document.

Naccache, D. and Stern, J. (1998). A new public key cryptosystem based on higher residues, *CCS '98: Proceedings of the 5th ACM conference on Computer and communications security*, ACM Press, San Francisco, California, USA, pp. 59–66.

Naor, M. and Pinkas, B. (1999). Oblivious transfer and polynomial evaluation, *Proceedings of the Annual ACM Symposium on Theory of Computing*, Atlanta, Georgia, USA, pp. 245–254.

Naor, M. and Pinkas, B. (2001). Efficient oblivious transfer protocols, *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, Washington, DC, USA*, pp. 448–457.

Naor, M., Pinkas, B. and Sumner, R. (1999). Privacy preserving auctions and mechanism design, *in Proceedings of the 1st ACM Conference on Electronic Commerce*, pp. 129–139.

Okamoto, T. and Uchiyama, S. (1998). A new public-key cryptosystem as secure as factoring, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland*, pp. 308–318.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic*, pp. 223–238.

Paillier, P. and Pointcheval, D. (1999). Efficient public-key cryptosystems provably secure against active adversaries, *Advances in Cryptology - ASIACRYPT '99, International Conference on the Theory and Applications of Cryptology and Information Security, Singapore*, pp. 165–179.

Pinkas, B. (2002). Cryptographic techniques for privacy-preserving data mining, *SIGKDD Explorations* **4**(2): 12–19.

Pinkas, B., Schneider, T., Smart, N. P. and Williams, S. C. (2009). Secure two-party computation is practical, *IACR Cryptology ePrint Archive* **2009**: 314.

Prasad, P. K. and Rangan, C. P. (2006). Privacy preserving birch algorithm for clustering over vertically partitioned databases, *Secure Data Management,third VLDB Workshop, SDM 2006*, Seoul, Korea, pp. 84–99.

Quinlan, J. R. (1986). Induction of decision trees, *Machine Learning* **1**(1): 81–106.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Rabin, M. O. (1981). How to exchange secrets by oblivious transfer, *Technical report*, Technical Report TR-81, Aiken Computation Laboratory, Harvard University.

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

Rizvi, S. and Haritsa, J. (2002). Maintaining data privacy in association rule mining, *Proceedings of the 28th Conference on Very Large Data Base*, Hong Kong, pp. 682–693.

Rud, O. P. (2001). *Data mining cookbook : modeling data for marketing, risk and customer relationship management*, Wiley.

Ryger, R. S., Kardes, O. and Wright, R. N. (2008). On the lindell-pinkas secure computation of logarithms: From theory to practice, *Practical Privacy-Preserving Data Mining*, pp. 21–29.

Safavian, S. and Landgrebe, D. (1991). A survey of decision tree classifier methodology, *IEEE Trans. Systems, Man and Cybernetics* **21**(3): 660–674.

Sakuma, J., Kobayashi, S. and Wright, R. N. (2008). Privacy-preserving reinforcement learning, *Proceedings of the 25th international conference on Machine learning*, ACM, New York, USA, pp. 864–871.

Samarati, P. (2001). Protecting respondents' identities in microdata release, *IEEE Trans. Knowl. Data Eng.* **13**(6): 1010–1027.

Schoenmakers, B. and Tuyls, P. (2006). Efficient binary conversion for paillier encrypted values, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia*, pp. 522–537.

Seifert, J. W. (2007). Data mining and homeland security: An overview, CRS Report for Congress.

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

Sheikholeslami, G., Chatterjee, S. and Zhang, A. (1998). Wavecluster: A multi-resolution clustering approach for very large spatial databases, *Proceedings of 24th International Conference on Very Large Data Bases, VLDB*, New York, USA, pp. 428–439.

Shmueli, G., Patel, N. R. and Bruce, P. C. (2006). *Data mining for business intelligence : concepts, techniques, and applications in Microsoft Office Excel with XLMiner*, Wiley-Interscience.

Smart, N. P. and Vercauteren, F. (2010). Fully homomorphic encryption with relatively small key and ciphertext sizes, *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France*, pp. 420–443.

Stehlé, D. and Steinfeld, R. (2010). Faster fully homomorphic encryption, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore*, pp. 377–394.

Strassen, V. (1969). Gaussian elimination is not optimal, *Numerische Mathematik* **13**(4): 354–356.

Su, C., Bao, F., Zhou, J., Takagi, T. and Sakurai, K. (2007). Privacy-preserving two-party k-means clustering via secure approximation, *21st International Conference on Advanced Information Networking and Applications (AINA 2007), Workshops Proceedings, Volume 1, Niagara Falls, Canada*, pp. 385–391.

Tan, P.-N., Steinbach, M. and Kumar, V. (2006). *Introduction to data mining*, Pearson Addison Wesley.

Teng, Z. and Du, W. (2007). A hybrid multi-group privacy-preserving approach for building decision trees, *The 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2007)*, Nanjing, China, pp. 296–307.

Teo, S. G., Han, S. and Lee, V. C. S. (2013). Privacy preserving support vector machine using non-linear kernels on hadoop mahout, *16th IEEE International Conference on Computational Science and Engineering, CSE 2013, Sydney, Australia*, pp. 941–948.

Vaidya, J. and Clifton, C. (2002). Privacy preserving association rule mining in vertically partitioned data, *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, pp. 639–644.

Vaidya, J. and Clifton, C. (2003). Privacy-preserving k-means clustering over vertically partitioned data, *Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining*, Washington, DC, USA, pp. 206–215.

Vaidya, J. and Clifton, C. (2004). Privacy preserving naïve bayes classifier for vertically partitioned data, *Proceedings of the SIAM International Conference on Data Mining*, Lake Buena Vista, Florida, USA, pp. 522–526.

Vaidya, J. and Clifton, C. (2005). Secure set intersection cardinality with application to association rule mining, *Journal of Computer Security* **13**(4): 593–622.

Vaidya, J., Clifton, C., Kantarcioglu, M. and Patterson, A. S. (2008). Privacy-preserving decision trees over vertically partitioned data, *ACM Transactions on Knowledge Discovery from Data (TKDD)*, pp. 14:1–14:27.

Vaidya, J., Kantarcioglu, M. and Clifton, C. (2008). Privacy-preserving naïve bayes classification, *Journal of Very Large Data Bases* **17**(4): 879–898.

Vaidya, J., Yu, H. and Jiang, X. (2008). Privacy-preserving svm classification, *Knowledge and Information Systems* **14**(2): 161–178.

van Dijk, M., Gentry, C., Halevi, S. and Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, France*, pp. 24–43.

Verykios, V. S., Bertino, E., Fovino, I. N., Provenza, L. P., Saygin, Y. and Theodoridis, Y. (2004). State-of-the-art in privacy preserving data mining, *SIGMOD*, pp. 50–57.

Verykios, V. S., Elmagarmid, A. K., Bertino, E., Saygin, Y. and Dasseni, E. (2004). Association rule hiding, *IEEE Trans. Knowl. Data Eng.* pp. 434–447.

Veugen, T. (2014). Encrypted integer division and secure comparison, *International Journal of Applied Cryptography*, pp. 166–180.

Wang, W., Yang, J. and Muntz, R. R. (1997). Sting: A statistical information grid approach to spatial data mining, *Proceedings of Twenty-Third International Conference on Very Large Data Bases*, Morgan Kaufmann, Athens, Greece, pp. 186–195.

Wright, R. and Yang, Z. (2004). Privacy-preserving bayesian network structure computation on distributed heterogeneous data, *Proceedings of the 10th ACM International Conference on Knowledge Discovery and Data Mining*, Seattle, WA, USA, pp. 713–718.

Yao, A. C. (1986). How to generate and exchange secrets (extended abstract), *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada*, pp. 162–167.

Yi, X., Paulet, R. and Bertino, E. (2014). *Homomorphic Encryption and Applications*, Springer Briefs in Computer Science, Springer.

Yu, H., Jiang, X. and Vaidya, J. (2006). Privacy-preserving SVM using nonlinear kernels on horizontally partitioned data, *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), Dijon, France*, pp. 603–610.

Yu, H., Vaidya, J. and Jiang, X. (2006). Privacy-preserving svm classification on vertically partitioned data, *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Singapore, pp. 647–656.

Zhan, J., Matwin, S. and Chang, L. (2007). Privacy-preserving collaborative association rule mining, *Journal of Network and Computer Applications* **30**(3): 1216–1227.

Zhang, N., Wang, S. and Zhao, W. (2004). A new scheme on privacy preserving association rule mining, *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Pisa, Italy, pp. 484–495.

Zhang, N., Wang, S. and Zhao, W. (2005). A new scheme on privacy-preserving data classification, *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, Chicago, Illinois, USA, pp. 374–383.

Zhang, T., Ramakrishnan, R. and Livny, M. (1996). Birch: An efficient data clustering method for very large databases, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada*, pp. 103–114.

Zhong, S. (2007). Privacy-preserving algorithms for distributed mining of frequent itemsets, *Inf. Sci.* **177**(2): 490–503.

Zhou, Y., Guo, Q. and Gan, R. (2009). Improved aco algorithm for resource-constrained project scheduling problem, *International Conference on Artificial Intelligence and Computational Intelligence, AICI'09*, pp. 358–365.