AN ONLINE CONSECUTIVE SECURE MULTI PARTY COMPUTATION ALGORITHM FOR DISTRIBUTED PRIVACY PRESERVING FREQUENT ITEMSET MINING

ABDOLREZA RASOULI KENARI

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Doctor of Philosophy (Computer Science)

Faculty of Computer Science and Information System
Universiti Teknologi Malaysia

MARCH 2011

DECLARATION

I declare that this thesis entitled “An Online Consecutive Secure Multi Party Computation Algorithm For Distributed Privacy Preserving Frequent Itemset Mining” is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

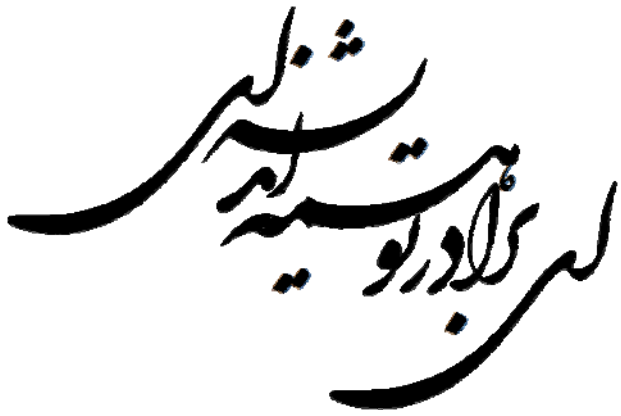
Signature	:	.....
Name	:	Abdolreza Rasouli
Date	:	1 March 2011

DEDICATION

O' brother, you are all (in Fact) Thought

The rest, you are the Bones and Meat

Jalal al-Din Rumi



To my beloved mother and father

to my loving wife

and Ilia

ACKNOWLEDGEMENT

In the name of God, the Most Gracious, the Most Merciful. I would like to express my deep and sincere gratitude to my supervisor, Professor Dr Mohd Aizaini Maarof and in memory of Dr Mohd Nor Mohd Sap (God bless him). Their wide knowledge and their logical way of thinking have been of great value for me. Their understanding, encouraging and personal guidance have provided a good basis for the present thesis.

I also wish to thank Meghdad PhD of English education, for revising the English of my manuscript. During this work I have collaborated with many colleagues for whom I have great regard, and I wish to extend my warmest thanks to all those who have helped me with my work in the Faculty of Computer Science and Information Systems, University of Technology, Malaysia.

I owe my loving thanks to my wife Mahboubeh, my son Illia and our parents. They have lost a lot due to my research abroad. Without their encouragement and understanding it would have been impossible for me to finish this work. My special gratitude is due to my family, my sister and brothers, their families for their loving support.

ABSTRACT

Every day large volume of information produces and stores among multi parties systems. Although these data are produced by companies unrelated to each other and are stored in various parties, but when they are gathered together much valuable information and patterns reveals. Data mining over distributed data discovers this costly knowledge. However, ownership of data by different companies and maintain confidentiality of data is the main challenge. Secure Multi Party Computation is a set of methods that perform mathematic computation over the multi party distributed data with ensuring the privacy preserving of the confidential data. Most of these methods use a shared secret key to ensure the privacy of each party. All parties should be present to share the secret keys, but unfortunately in many real cases, the parties are not joining the process at the start time. The aim of this research is to design a new innovative online consecutive secure multi party computation algorithm is purposed in case of online parties. The main problems addressed in this research are the online secret sharing and the consecutively two party computation. The infinite convergent product sequences are employed to overcome the dependency problem between the shared secret key and users' public keys, which make the algorithm runs offline. The designed online secret sharer allows the parties to join the system during process life. The second problem is cleared by adding a two party computation randomizer to the system. The two party randomizer ensures the privacy of the online consecutive computation. The designed algorithm is tested over the E-Voting and web polls case studies and the result proves the robustness, security and applicability of the algorithm. Moreover, a distributed online frequent itemset mining is developed using the purposed algorithm and the result demonstrates the performance, efficiency and practicability of the multi party computation algorithm which is considered not enough by the researchers.

ABSTRAK

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	DECLARATION	**II**
	DEDICATION	**III**
	ACKNOWLEDGEMENT	**IV**
	ABSTRACT	**V**
	ABSTRAK	**VI**
	TABLE OF CONTENTS	**VII**
	LIST OF TABLES	**XII**
	LIST OF FIGURES	**XIII**
	LIST OF ALGORITHMS	**XV**
	LIST OF ABBREVIATIONS	**XVI**
1	**INTRODUCTION**	**1**
	1.1 Overview	1
	1.2 Problem Background	2
	1.3 Statement of Problem	4
	1.4 Research Goal	5
	1.5 Research Objectives	5
	1.6 Research Scope	6
	1.7 Research Contribution	7
	1.8 Research Justification	7
	1.9 Thesis Organisation	9
2	**LITERATURE REVIEW**	**11**
	2.1 Introduction	11

2.2	Security and Privacy Model Definition	12
	2.2.1 Definitions of Privacy	13
	2.2.2 Privacy and Security Models	13
2.3	Multi Party Computation	15
	2.3.1 Secure Multi Party Computation Definition	16
	2.3.2 Two Party Computation	16
2.4	Secure Multi Party Computation Solutions	18
2.5	Encryption Based Solution	21
	2.5.1 Oblivious Transfer	21
	2.5.2 Secret Sharing	21
	2.5.3 Threshold Encryption	23
	2.5.4 Homomorphic Encryption	24
	2.5.5 ElGamal Cryptosystem	24
	2.5.6 Unpadded RSA	25
	2.5.7 Paillier Cryptosystem	26
	2.5.8 Encryption Based Solution Advantages and Disadvantages	27
2.6	Randomization and Transformation Based Solutions	27
	2.6.1 Data Perturbation	27
	2.6.2 Data Distortion	28
	2.6.3 Randomization Based Solution Advantages and Disadvantages	30
2.7	Trusted Third Party	30
	2.7.1 Online Semi-trusted Third Party (STTP)	31
	2.7.2 Offline Semi-trusted Third Party (STTP)	32
	2.7.3 Trusted Third Party Advantage and Disadvantages	32
2.8	Multi Party Application	33
	2.8.1 Privacy Preserving Co-operative Scientific	35
	2.8.2 Privacy Preserving Database Query	36
	2.8.3 Privacy Preserving Intrusion Detection	36
	2.8.4 Privacy Preserving Data Mining	37
	2.8.5 Privacy Preserving Geometric Computation	38

2.8.6 Privacy Preserving Statistical Analysis: 38
2.8.7 E-Voting 39
2.9 Secure Distributed Frequent Itemset Mining 40
2.10 Privacy Preserving Frequent Itemset Mining 41
2.10.1 Association Rule Mining 43
2.10.2 Apriori Algorithm 45
2.10.3 FP-Growth Algorithm 46
2.10.4 Eclat Algorithm 49
2.10.5 Frequent Itemset Mining Algorithms Advantage an Weaknesses 51
2.10.6 Distributed Apriori algorithm 53
2.11 Summary 55

3 RESEARCH METHODOLOGY 56
3.1 Introduction 56
3.2 Research Motivation 57
3.3 Research Framework 59
3.3.1 Online Parameters for Distributed Secure Multi Party Computation 60
3.3.2 Online Secret Sharing 62
3.3.3 Online Consecutive Computation 63
3.3.4 Distributed Frequent Itemset Mining 65
3.4 Summary 68

4 A NEW ONLINE CONSECUTIVE SECURE MULTI PARTY COMPUTATION ALGORITHM 69
4.1 Introduction 69
4.2 ElGamal Enhanced Multi Party Computation Algorithm 70
4.2.1 Algorithm Preliminaries 71
4.2.2 Enhanced Offline Algorithm 72
4.2.3 Enhanced Offline Algorithm Demonstration 74
4.3 Online Secret Sharing 75

4.3.1 Algorithm Preliminaries 77
4.3.2 Online Secret Sharing algorithm 78
4.3.3 Online Secret Sharing Algorithm Demonstration 81
4.4 Consecutive Multi Party Computation 82
4.4.1 Algorithm Preliminaries 83
4.4.2 Online Consecutive Algorithm 85
4.4.3 Initializing Setup 86
4.4.4 Joining and Encrypting Data 87
4.4.5 Computing the Randomized Production 87
4.4.6 Computing the Aggregation Outcome 88
4.4.7 Online Consecutive Algorithm Demonstration 91
4.4.8 Algorithm Security Analysis 94
4.5 Distributed Frequent Itemset Mining 94
4.5.1 Algorithm Preliminaries 95
4.5.2 Privacy Preserving Distributed Frequent Itemset 98
4.6 Summary 101

5 E-VOTING AND DISTRIBUTED FREQUENT ITEMSET MINING, EXPRIMENTAL RESULTS AND CASE STUDIES 103
5.1 Introduction 103
5.1.1 Test Environment Architecture 103
5.2 E-Voting and Web Polls 104
5.2.1 Offline E-Voting Experimental Result 105
5.2.2 E-Voting Experimental Result Using the Online Secret Sharing 109
5.2.3 Online Consecutive E-Voting; Experimental Result 112
5.3 Distributed Frequent Itemset Mining Experimental Result 117
5.4 Summary 122

6 CONCLUSION 123

6.1 Introduction 123

6.2 Evaluation of the Designed Algorithm 124

6.3 Thesis Contribution 127

6.4 Further Work 128

REFERENCES 129

LIST OF TABLES

TABLE NO.	TITLE	PAGE
Table 2.1	Example data base with 4 items and 5 transactions	44
Table 2.2	Sample Transactions Database	48
Table 2.3	Association rule mining algorithm comparison	52
Table 4.1	Example of the Boolean itemset matrix	96
Table 5.1	Processing time for different number of voters	106
Table 5.2	Processing Time of Online Secret Sharing Algorithm	110
Table 5.3	Processing Time of Online Consecutive Algorithm	115
Table 5.5	Execution time in millisecond related to the number of transactions	117

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
Figure 2.1	The Security Model	15
Figure 2.2	Effect of secure multi party computation solutions	19
Figure 2.3	Secure multi party computation solutions	20
Figure 2.4	Offline Secret Sharing	22
Figure 2.5	Data mining domain; the research focus is bolded	41
Figure 2.6	Example of an FP-tree	49
Figure 2.7	Ranking of Each Algorithm Running over All Datasets	52
Figure 2.8	Average Timing for Each Algorithm	53
Figure 3.1	Research motivation	58
Figure 3.2	Research Framework	59
Figure 3.3	Research Procedure	60
Figure 3.4	Enhancing offline multi party computation procedure	61
Figure 3.5	Online Secret Sharing	62
Figure 3.6	Designing online secret sharing procedure	63
Figure 3.7	Online consecutive computation procedure	65
Figure 3.8	Distributed Frequent Itemset Mining	66
Figure 3.9	Frequent Itemset Mining over Two Vertically Partitioned Databases	67
Figure 3.10	Secure distributed frequent itemset mining procedure	67
Figure 4.1	System Architecture	71
Figure 4.2	ElGamal based MPC procedure	73
Figure 4.3	Offline secret sharing scheme	76
Figure 4.4	Online secret sharing scheme	77
Figure 4.5	Consecutive computation reveals the confidential data	84
Figure 4.6	Online Consecutive Computation Model	85

Figure 4.7	Frequent itemset mining with online consecutive secure multi party computation; Model and framework	95
Figure 5.1	E-Voting using offline secret sharer	105
Figure 5.2	Total time of E-Voting process related to Number of respondents	106
Figure 5.3	Total time related to Number of Voters and Range of V	107
Figure 5.4	Overall response time of the proposed homomorphic methods	108
Figure 5.5	Key generation time and package size	109
Figure 5.6	Total time of Algorithm execution related to the Number of respondents	111
Figure 5.7	Total time related to Number of Voters and Range of V – 3d view	112
Figure 5.8	Online E-Voting process	113
Figure 5.9	The online consecutive processing time related to number of voters	115
Figure 5.10	The percentage portion of each step among the total processing time	116
Figure 5.11	Execution time related to the number of transactions	119
Figure 5.12	Execution time related to the number of parties	121

LIST OF ALGORITHMS

ALGORITHM NO.	TITLE	PAGE
Algorithm 2.1	Apriori Algorithm	46
Algorithm 2.2	Pseudo code of insert_tree (p | P, R)	48
Algorithm 2.3	Basic Eclat algorithm	51
Algorithm 2.4	Distributed Apriori algorithm	55
Algorithm 4.1	ElGamal Based offline multi party computation algorithm	74
Algorithm 4.2	Online secret sharing algorithm	80
Algorithm 4.3	Secure online consecutive multi party computation algorithm	90
Algorithm 4.4	Privacy Preserving Distributed Frequent Itemset Algorithm	101

LIST OF ABBREVIATIONS

ABBREVIATION		DEFINITION
MPC	-	Multi Party Computation
2PC	-	Two Party Computation
SCET	-	Secure Computing, Economy and Trust
SIMAP	-	Secure Information Management and Processing
SMC	-	Secure Multi-Party Computation
TTP	-	Trusted Third Party
STTP	-	Semi Trusted Third Party
PPCG	-	Privacy Preserving Computational Geometry
PIR	-	Private Information Retrieval
SS	-	Secret Sharing
FIM	-	Frequent Itemset Mining
FIMI	-	Frequent Itemset Mining Implementation
IS	-	Itemset

CHAPTER 1

INTRODUCTION

1.1 Overview

Multi Party Computation (MPC) is a mathematical cryptographic technique that securely computes a function over distributed data among two (2PC) or number of parties. It is secure and preserves the privacy of parties' data and do not reveal the sensitive data except the function's result. Multi party computations use a shared secret which is apportioned between all parties to ensure the privacy of the data. The shared secret guarantees that the function only is computable over all parties. The method of sharing the secret makes the algorithms to behave offline and inappropriate for online problems. In this research, the online secret sharing problem is solved by convergent infinite product sequences. Another problem of online multi party computation is the leaking information between two consecutive computations, which is solved by two party computation techniques. The purposed innovative online consecutive multi party computation ensures the security and privacy of all parties' owned sensitive data.

The algorithm is highly practicable for distributed data mining. In fact, today, the huge amount of data will be used among the large number of companies. There exist a lot of valuable patterns and roles, which are hided between voluminous data. Data mining tools have been developed to discover these worth facts. Companies can learn of hidden patterns and rules in their joint databases by using the distributed data mining tools to predict the future of their business. While information is distributed between two (or more) companies, and each corporation owns a portion of

information, they should collaborate with themselves to jointly mine the combined information and find fascinated pattern and rules that they are interested. This scenario is known as distributed data mining. It is sophisticated where the data has been distributed among the number of parties because of privacy of their sensitive data. Each company leans to gain the advantage of data mining tools, but he does not propend to share his sensitive data. However, all parties are interested to the result of data mining process over joint databases and would like to participate, but privacy issues may avoids them for exposing their own part of data to other parties. Multi party computation solves this conflict. The novel purposed algorithm ensures the privacy preserving of data using cryptographic tools as well as precious aggregation over the distributed data.

1.2 Problem Background

The secure multi-party computation also known as (MPC) is one of the principal results of the theory of cryptography. First, Yao (Yao, 1982, Yao, 1986) introduced the multi-party computation and nowadays many authors have attended many optimizations and extensions to the basic concept, for two main branch; the two-party (2PC) and the multi-party (MPC) computation (Canetti, 2000, Goldreich, 2004, Goldreich *et al.*, 1987, Jarecki and Shmatikov, 2007, Lindell and Pinkas, 2007, Mohassel and Franklin, 2006, Woodruff, 2007). Most of recently papers on secure multi-party computation area have been focused on theory of multi-party computation and there is no much applicable implementing of MPC, although, in the few last years some practical implementation of multi-party computation has been appeared (Assaf *et al.*, 2008, Bogetoft *et al.*, 2006, Dahlia *et al.*, 2004, Peter *et al.*, 2009, Yehuda *et al.*, 2008).

Secure multi-party computation divides into two main approaches (Pinkas *et al.*, 2009). he first way is based an arithmetic circuit representation of the computed function and secret sharing, such as in the BGW (Ben-Or, Goldwasser and Wigderson) or CCD (Chaum, Crepeau and Damg°ard) protocols (Chaum *et al.*, 1988, Michael *et al.*, 1988). While most of the participants are trusted (the protocol is not

working in two parties case), this method usually applies. Another way is based upon a binary circuit. The approach is designed especially in case of original two-party garbled circuit construction of Yao (Yao, 1986), and in the GMW (Goldreich, Micali and Wigderson) multi-party protocol (Goldreich *et al.*, 1987).

Due to the nature of secret sharing, most of the existing multi party computation algorithms are offline and the rest are semi-online. There are two main problems to change the offline multi party computation algorithms to an online mode. The first problem is how a secret could share online among all parties. The shared secret is usually a calculation of the all parties' keys and therefore, all parties should present before the process starts to share their secret key. It means that no new party can join the system after the process started. The offline secret sharing problem avoids the algorithm to be applicable in online cases such as online e-voting, e-bidding or web polls.

The second problem is the consecutive computation which means that some sensitive information reveals by observing the two consequent computation results. In case of online multi party computation, if an adversary obtains two consequent computation results, a simple calculation leads to find the *i*th party's private data. The calculation also does not need to decrypt the encrypted posted data, knowledge about the public or private keys, shared secret key or even any collaboration with some parties.

Although the potential of multi party computation for solving many issues is obvious, but there is no much practical application is reported in this area. Secure Multi-Party Computation (SMC) computes the result of function upon the private information of parties with the related environment, ensuring the minimum exposure (Mishra and Chandwani, 2007). SMC provides arguable solutions for organizations for problems like privacy-preserving database query, privacy-preserving scientific computations, privacy preserving intrusion detection and privacy-preserving data mining. Privacy preserving data mining protects the secrecy of raw records while the distributed data mining is processed over aggregate data. The main problem which is addressed by researchers is how to mine the accumulated data from different

organizations with holding the confidentially of each party. For instance, how two (or more) business can discover frequent itemsets among their different databases without revealing their raw records to the others or how an online auction can find the best market clearing price which is equal to best tradeoff between supplies and demands without revealing the customers' online bids or even how the web page can compute the result of the web page poll without revealing the visitors' opinion to the web page owner. However, recently, the privacy preserving data mining has been attended, but an efficient algorithm for online data mining has been not reported.

1.3 Statement of Problem

In this study, we intend to come up with an approach effectively to solve the problem of online consecutive multi party computation algorithm that overcomes the 2PC problems. The research question is:

"How to produce a secure online consecutive cryptic multi party computation algorithm that shares the secret online and solves the 2PC problem?"

In order to answer the main issue raised above, the following issues need to be addressed as a prerequisite:

- What is the definition of security, reliability and privacy?
- How the cryptography could synthesize and which cryptography technique is more suitable?
- What do we mean from online term?
- What is the problem of offline multi party computation algorithms?
- How do we share the multi party computation secrets online?

- The main problem will appear in online a case is the 2PC problem. How do we overcome the 2PC problem?
- Whether the purposed algorithm can be combined with data mining techniques?

1.4 Research Goal

The goal of this study is to develop a novel secure, privacy preserver, reliable and online crypto multi party computation algorithm that can compute an aggregation over distributed data and to ensure that no sensitive information reveal even in case of 2PC scenario. The minor aim is to apply the innovative algorithm in data mining purposes.

1.5 Research Objectives

In order to achieve the aforementioned aim, listed below are the objectives of this thesis:

- To enhance the offline multi party computation algorithms supporting speed, easy key management and network low overhead for online purposes.
- To share the multi party computation secret key in online form.
- To overcome the leaking sensitive information problem in term of the consecutive 2PC problem.
- To ensuring the applicability of the algorithm with admissible response time and developing data mining tool using the purposed algorithm to prove the usability of the algorithm.

1.6 Research Scope

This research is significantly involved with secure multi party computation algorithms. The existing algorithms need to share a secret between all parties, so they need to gather all parties before computation starts. Due to secret sharing issue, current multi party computation algorithms work at offline mode. A secure, applicable and standard offline multi party aggregation algorithm is developed using ElGamal cryptosystem, due to its simplicity in key management and its exponential feature. The ElGamal enhanced aggregation algorithm will be discussed in Chapter 4.3.

A mathematical fact of the convergent infinite product series is revised to sharing the secret among the parties as the online form. The fact is that the result of the convergent infinite series will be invariant while the number of term increases. The invariant convergence value is used for shared secret and then each term of the series is used by parties for sending data. An online multi party computation algorithm using the convergent infinite product sequence is developed and will be detailed in Chapter 4.4.

While a function computes among some parties online, it means that the result could be computed any time during the process. By this assumption, a foible occurs. If an adversary computes the result of *i*th step and (*i*+1)th step, then the difference between these two values equal to owned value of the $(i+1)^{th}$ person, despite of cryptographic method, without decrypting the values and even without knowledge about the keys. The Yao's secure two party computation (Yao, 1982) as the state of the art in the multi party computation field is selected as the base of the solution. The online multi party computation algorithm is combined with a new two party computation technique for avoiding adversaries to earn sensitive data. The algorithm ensures the privacy of sensitive data and defeats the two party computation attacks. Chapter 4.5 describes the details of the purposed innovative algorithm.

The essence of multi party computation is finding the result of a function over distributed data without learning about the data or what the exact values of the data

are. On the other hand, in data mining also the exact values of data do not important, but a pattern that hides among the data is important. In distributed data mining privacy concern avoids the parties to join to the data mining process. With the above view of multi party computation, the best solution in case of distributed data mining is using multi party computation, which is not considered by researchers. In Chapter 5, the association rule miming, especially frequent itemset mining is selected to impalement with the purposed algorithm. The developed tools is solved the big problem of distributed data mining and also shows the practicality of the algorithm and emerges the researchers to work more in the new scope known as online distributed data mining using multi party computation.

1.7 Research Contribution

According to research objectives, the research contributions are listed below;

1. Develop an ElGamal based offline multi party computation algorithm with high-speed and easier to use.
2. Creating the online secret sharing process using convergent infinite product series to produce the online multi party computation algorithm.
3. Improving the online aggregation algorithm using two party computation techniques and dominate the sensitive information revealing problem.
4. Applying the innovative algorithm to data mining tools and ensure the applicability of the algorithm.

1.8 Research Justification

Although, the multi party computation was introduced three decades ago by

Yao (Yao, 1982), but few practical applications of multi party computation is reported. However, the potential of the multi party computation for solving the problems is obvious, but the researchers do not interest to use multi party computation in real practical applications. It is probably due to the fact that the implementation of first generation multi party computation technique was not enough efficient. Another important factor was poorly understood of the multi party computation potentials (Peter *et al.*, 2009). Many researchers try to do improve the efficiency of the multi party computation techniques, recently (Damgård and Nielsen, 2003, Damgård and Thorbek, 2007, Gennaro *et al.*, 1998). Although, in the few last year some practical implementation of multi-party computation has been appeared (Amirbekyan and Estivil-Castro, 2007, Bogetoft *et al.*, 2006, Ma and Sivakumar, 2005, Ma and Sivakumar, 2006, Yehuda *et al.*, 2008).

A recent approach is focused on application of multi party computation in a range of economic purposes, which are especially engrossing for practical use (Peter *et al.*, 2009). They involved in two economic research projects: SCET (Secure Computing, Economy and Trust) and SIMAP (Secure Information Management and Processing), because of the important role of trusted third party in the economic research field. The application of multi party computation is to find the best Market Clearing Price which means the price of per merchandise that is dealt. If assume that when the price increase, supply will raise and demand will reduce, the auctioneer looks for a price where supplies are equal to demands. Secure multi party computation is employed to hold the privacy of bids that should be computed.

The applicability of multi party computation which is concerned in this research is utilising the multi party computation in distributed data mining tools. One of the most important tasks in data mining application is mining frequent itemset. These applications include the discovery of association rules, strong rules, correlations, sequential rules, episodes, multi dimensional patterns, and several other important discovery tasks. Association rules are widely used in various areas such as telecommunication networks, market and risk management, and inventory control. Association rules also are employed today in many application areas, including Web usage mining, intrusion detection and bioinformatics. In case of distributed data, the

privacy of sensitive data avoids the miner to find frequent itemsets among the partitioned data. Multi party computation is the best technique that can solve the privacy preserving issues in distributed data mining approaches.

1.9 Thesis Organisation

The thesis consists of six chapters. Each chapter is briefly described as follows:

1. Chapter 1 describes the background, statement of the problem, aim, objectives, research framework, scope, thesis organisation and ended with the thesis contributions.

2. Chapter 2 illustrates the full literature review of existing methods in offline multi party computation and their weaknesses and advantages. The chapter continues with a detail of preliminary requirements cryptographic techniques and describing the existing distributed data mining tools.

3. Chapter 3 briefly describes the research methodology, architecture framework and general overview of the research steps.

4. Chapter 4 describes a new purposed online consecutive secure multi party computation algorithm

    4.1. Section 4.1 starts the chapter with a brief introduction and propounding the main issues.

    4.2. Section 4.2 describes the implementation of the offline aggregation algorithm enhance by ElGamal cryptosystem as the base as the designed algorithm.

    4.3. Section 4.3 introduces a new online aggregation algorithm that uses an online secret sharing process using the math

convergent infinite product series and its weaknesses and advantages relying on the real case studies.

4.4. Section 4.4 presents the designed online consecutive multi party computation algorithm that overcomes the new emerged problem known as the 2PC issue.

4.5. Section 4.5 describes the developed secure distributed frequent itemset mining algorithm using designed method.

5. Chapter 5 deals with applying the novel algorithm in real case studies and investigates on experimented results

5.1. Section 5.1 starts with the explanation of the experimental result environment and technical specifications.

5.2. Section 5.2 discusses the e-voting process as the selected case study.

5.3. Section 5.3 deals with the result of the novel purposed algorithm in data mining tools.

6. Chapter 6 ends with a conclusion.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Volume of the data is extremely grown and with the growth of the internet, joint computation has increased considerably. In joint computation, the participants who have contributed the process wish to compute a function on their data. Slightly the problem can be solved this way; all the participants can send their inputs to a third party, who computes the required function on these inputs and gives back the output to each party. The main issue is the mutually distrusting and privacy. Specifically, what if the players are not willing to reveal their inputs but still wish to jointly compute some function. For example, suppose that the government wants to realize the number of the epidemic disease cases. The hospitals are requested to submit their statistics. However, the hospitals definitely would like to conceal the identity of the patients as it might lead to social dogma.

The Secure Multi Party Computation (SMPC), introduced by Yao (Yao, 1982) essentially aims to address such issues. He defined the problem as there are *n* participants who wish to jointly compute some function on their owned data, such that at the end of the evaluation, no party gains any additional information about the local input of any other party apart from what can be implied from the output. During the computation some parties may try to gain additional information that they should know. The mentioned parties are called adversary that may cooperate with some other parties to learn remained parties' data. A subset of *t* parties who cooperate each others is called *t*-adversary. Inverse, the party which faithfully executes the protocol

delegated to him and does no more, is called non-faulty party.

There exists a very rich literature on Secure Multi Party Computation. As the first solution to the problem, Goldreich *et al*. (Goldreich *et al.*, 1987) presented a protocol that allows n parties to securely compute an arbitrary function against a computationally bounded, active t-adversary if and only if $t < n / 2$. The problem has since been studied under various settings such as secure games with honest majority (Goldreich *et al.*, 1987), Multiparty computation with faulty majority (Beaver and Goldwasser, 1989), Fault-Tolerant Computation in the Full Information Model (Goldriech *et al.*, 1991), Multiparty Unconditionally Secure Protocols (Chaum *et al.*, 1988), Adversary characterization in SMPC (Hirt and Maurer, 1997), Efficient MPC against Adaptive Adversary (Cramer *et al.*, 1999) and Player Simulation and General Adversary Structure in MPC (Hirt and Maurer, 2000), giving both possibility and impossibility results. However, a large part of the work has focused on solving any function.

This chapter starts with a description of the various biometrics systems. It then followed by the explanation of the iris structure. The chapter is continued with iris recognition that includes iris image preprocessing (iris acquisition), iris image segmentation, Normalisation and iris feature extraction. The chapter ends with a summary of existing methods.

2.2 Security and Privacy Model Definition

In this section, first the privacy is defined and then various models which are used to describe the strictness of security are defined, the different models of user that can interact with the securer multi party computation protocol and the definition of privacy in regards to secure multi party computation.

2.2.1 Definitions of Privacy

Secure multi party computation computes the function *f* on the multi datasets while preserving the privacy of the individual datasets and the identities of the subjects. Clifton *et al*. (Clifton *et al.*, 2002) define privacy with two main categories;

1. Individual privacy
2. Corporate privacy.

For individual privacy, no identifiable data, x_n, should be revealed other than to the party that owns it, and the identities of the data subjects themselves should be protected. It is most important that the data subject cannot be linked to any given data item even if an individual subject's identity or a particular data item is leaked.

They additionally state that protecting individual data items may not be enough. It is also need to protect other parties learning from the entire data collection, this is defined as corporate privacy. Therefore the algorithm should limit the ability of either party to see any intermediate results during computation of *f* or derive aggregate information for datasets they do not own.

2.2.2 Privacy and Security Models

Based on the Goldreich (Goldreich, 1998) definition, there are two security models;

1. The ideal model
2. The real model

The ideal model (Bella, 2008) is based on the trusted third party (TTP) who performs the computation. Such that the trusted third party will process the data

securely, not reveal the individual datasets or collude with either party. The ideal model is a perfect implementation of the Trusted Third Party Protocol. The ideal model usually is the basis for comparing the security of any real model.

In the real model, any real secure multi party computation algorithm is said to be secure in respect to a certain adversarial model if any leakage of information that occurs from the real model could theoretically occur also from the ideal model given that particular adversary model (Goldreich, 1998, Goldwasser, 1997).

Goldreich (Goldreich, 2004) also defines two adversary models;

1. The semi-honest model
2. The malicious model.

In some papers the semi-honest model names the passive model and the malicious model calls the byzantine model (Goldwasser, 1997, Goldwasser and Micali, 2006).

A semi-honest or passive adversary follows the secure multi party computation protocol correctly; but he tries to find more information than he should gain. The semi-honest adversary attempts to learn from any immediate results during the computation and whatever sent openly. Anyway a passive adversary will not deviate from the algorithm.

Inversely, the malicious or byzantine adversary does not bind himself to follow the algorithm correctly. Goldreich (Goldreich, 2004) presents three behaviors that a malicious party can perform that cannot be prevented;

1. Refusing to take part in the protocol at all
2. Aborting the protocol at any stage
3. Altering their input to the protocol at any given time.

Since these three actions cannot be prevented a protocol is said to be secure in respect to the malicious adversary model if the malicious party can only do these three actions. All of these actions can be used by the malicious party to try and derive the input dataset for the other party or the intermediate result of the function. For example one party can provide a blank input and receive the intermediate result of the function on the other party's input alone (Lindell and Pinkas, 2000).

The security model is depicted in Figure 2.1.

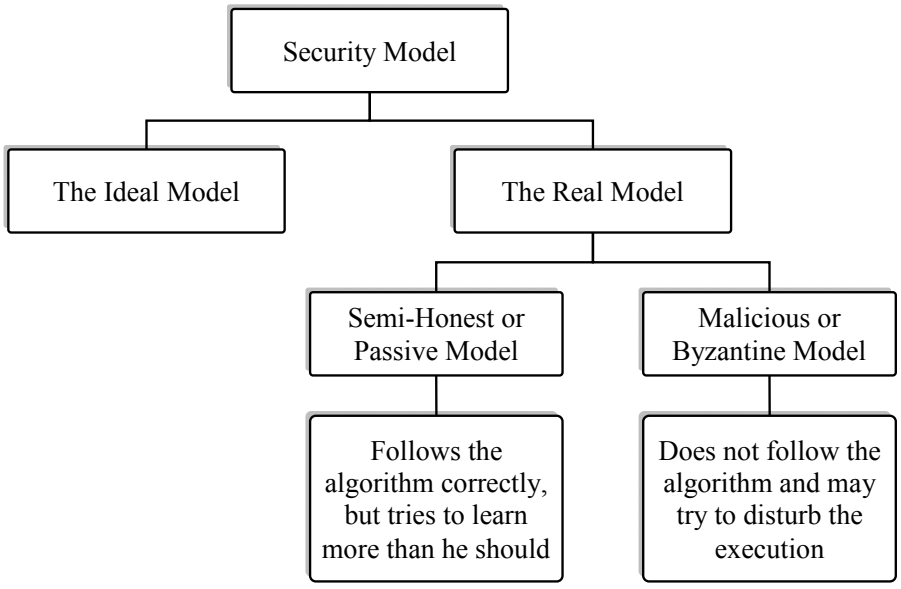


Figure 2.1 The Security Model

2.3 Multi Party Computation

The concept of the Secure Multi-party Computation (SMC) problem was introduced in 1982 by Yao (Yao, 1982). It can be defined as the requirement to compute any probabilistic function on any amount of inputs, for a distributed network of users where each user holds one set of inputs (Goldwasser, 1997). Additionally ensuring that the computation is carried out correctly and that no additional information is revealed to users than that which could be computed from a union of a particular user's input and output. This field is gaining youth interest; several research works in this area is being carried out by several scholars. Noted that

the privacy is vital concern for each organization and SMC protocols are a means to guarantee them in an easy manner.

2.3.1 Secure Multi Party Computation Definition

Suppose that there are n parties $P_1,P_2,\ldots,P_n$ that wish to perform a joint computation on their private data. Since, computation is to be performed on private data; it is important requirement that this private data should not be accessible to any other parties. Let $d_1,\ldots,d_n$ are the data corresponding to n organizations and let d_i is the data corresponding to ith party. It is required for computation that, d_i should not be accessible to any d_j where i ≠ j and j = 1,2,…,n. Therefore, each participant only gets the final results of joint computation $f(d_1,\ldots,d_n)$ without being aware of inputs involved and the computations made.

2.3.2 Two Party Computation

Most of all secure multi party computation supposes that the numbers of the parties are greater than two parties and mention it on their represented model. In term of two party computation the problem is different (Canetti *et al.*, 2002, Choi *et al.*, 2007). It is impossible to securely compute the determined algebraic function on the two parties' vital data without revealing it to the other party. Suppose that there are two party *A, B* and they own their vital data *x, y* respectively. The aim of the secure two party computation is to compute an algebraic determinate function $f(x,y) = x \oplus y$. If there is a secure way to compute $f(x, y)$, without knowing any knowledge about the secure computed method or cryptography technique which are used, the party *A* can compute the value of other party as $y = f(x,y) \ominus x$, where $\ominus$ denotes the inverse function of the employed algebraic determinate function. But with considering some conditions and in case of some relational or logical function, the two party computation is solvable in some specific problems.

Two-party protocols do not require a third party to perform the computation of the function, f, for them. Instead each party performs part of the function themselves and uses a variety of techniques to ensure the privacy of their input data.

Andrew Yao (Yao, 1982) introduced the problem and gave a solution to the millionaires problem, in which two millionaires with i and j millions ($1 < i, j < 10$) respectively want to compare their wealth and know who has got more wealth without revealing their actual wealth to the other person. One of the better solutions to Yao's millionaire's problem is give by Ioannis *et al*. (Ioannidis and Grama, 2003, Lin and Tzeng, 2005). Their computation and communication complexities are $O(d^2)$ both, where 2^d is the upper bound on their numbers which they want to compare.

Goldreich (Goldreich, 2008) defines the two-party protocol problem as a mapping via a random process, $f : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^* \times \{0, 1\}^*$, between a pair of inputs and a pair of outputs (one per party), such that the output-pair for an input-pair (x, y) is $f_1(x, y)$ and $f_2(x, y)$. The party holding x wants $f_1(x, y)$ and the party holding y wants $f_2(x, y)$. The secure version of the two-party protocol needs to achieve the computation of $f(x, y)$ without revealing y to the holder of x and x to the holder of y. To achieve this aim the two-party protocol must use encryption-based techniques or data distortion techniques to hide the input dataset for one party from the other party (Nielsen and Orlandi, 2009).

Recently the area of robust two-party computations in constant rounds has gained some attention. Specifically, the works of Jarecki and Shmatikov (Jarecki and Shmatikov, 2007), Lindell and Pinkas (Lindell and Pinkas, 2007) and Horvitz and Katz (Horvitz and Katz, 2007) gave protocols for two-party computation using Yao's garbled circuit that are secure against malicious adversaries. Jarecki and Shmatikov (Jarecki and Shmatikov, 2007) use a modified Camenisch-Shoup verifiable encryption scheme (Camenisch and Shoup, 2003) to allow the party that sends the garbled circuit to prove its correctness. Lindell and Pinkas (Lindell and Pinkas, 2000) use a cut-and-choose approach to proving security of Yao's garbled circuit against malicious adversaries and their method is more generic yet requires a few more rounds. Horvitz and Katz (Horvitz and Katz, 2007) showed a UC-secure protocol in

two rounds (four messages) using the DDH assumption. In their protocol, the two parties essentially run two instances of Yao's protocol simultaneously.

2.4 Secure Multi Party Computation Solutions

All the secure multi party algorithms proposed so far are categorized in two main classes (Wright, 2008);

1. Cryptographic Approach

   In this approach, the input from several parties is received in encrypted form by TTP.

2. Randomization Approach

   In this approach, the input from several parties is first concatenated and associated with a random number, in order to keep it secure.

In term of the cryptographic approach, the one-way functions play the main role of the secure multi party computation algorithm. The original protocol by Yao (Yao, 1986) also introduces the importance of one-way functions for the SMC problem. One-way functions first proposed by Diffie and Helman in 1976 (Diffie and Hellman, 1976). They are important for the SMC problem because represent a function whose input cannot be derived from the given output.

In the general manner, due to the high complexity, using the cryptographic techniques lead to low speed and inefficiency of the secure multi party computation algorithms. On the other hand, employing the randomization techniques decrease the accuracy of the algorithm. It is obvious that without the cryptographic or randomization technique, the privacy wastes. These issues are shown in the Figure 2.2, which denotes that using the cryptographic techniques increase the inefficiency and employing the randomization raise the inaccuracy of the protocol. Finally, lack

of both techniques goes to privacy losing.

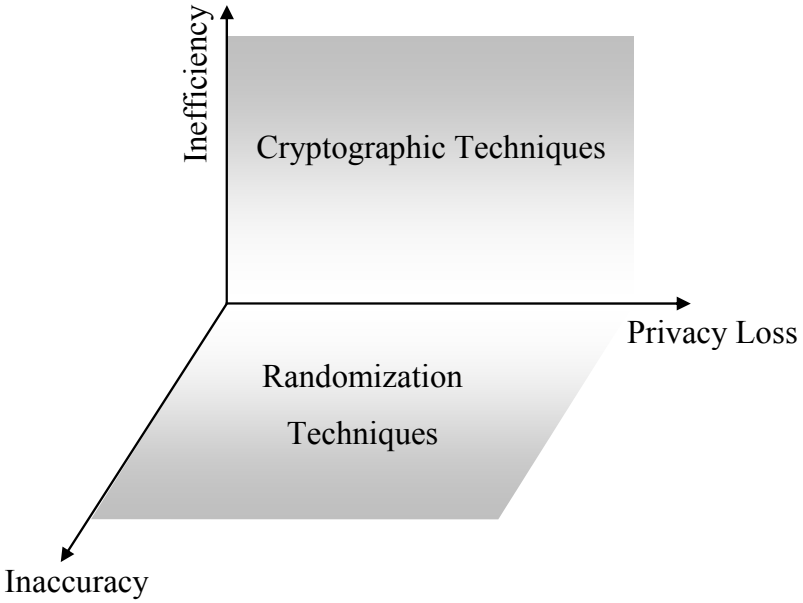


Figure 2.2 Effect of secure multi party computation solutions

Both Yao (Wright, 2008) and Goldreich *et al*. (Goldreich, 2008) agree that the secure multi party computation problem is solvable in theory but Goldreich (Goldreich, 2004) points out that a general solution to particular secure multi party computation problems will likely be impractical in terms of efficiency and therefore specific solutions should developed case-by-case. This is due to the fact that the speed of a general solution such as the Circuit Evaluation Protocol (Pinkas *et al.*, 2009), which represents the function *f* as a combinatorial circuit is based upon the size of the circuit and the size of the circuit required increases with the size of the input datasets. Notes that, the size of the circuit for a naïve implementation of a multiplication circuit is quadratic in regards to number of inputs; meaning that it will not produce an efficient solution.

There are several proposed solutions to secure multi party computation such as The Trusted Third Party (Du *et al.*, 2004, Du and Zhan, 2002, Goldreich, 1998, Goldreich, 2005, Peter *et al.*, 2009), The Oblivious Protocol (Rabin, 1981, Schaffner, 2010), 1-Out Of N Oblivious Protocol (Aiello *et al.*, 2001, Laur and Lipmaa, 2006, Naor and Pinkas, December 2000, Tzeng, 2002, Wong and Kim, 2010), Oblivious Evaluation Of Polynomials (Naor and Pinkas, 1999, Naor and Pinkas, 2001),

Threshold Cryptography (Damgård and Nielsen, 2003, Gennaro *et al.*, 1998, Wong and Kim, 2010) and Secret Sharing (Blundo and Masucci, 1999, Gennaro *et al.*, 1998, Li *et al.*, 2010, Shamir, 1979). Some other techniques like Secure Sum, Secure Set Union, Secure Size of Set intersection, Scalar Product, EM clustering can also be employed along with above mentioned approaches to find the SMC solutions (Bagüés *et al.*, 2010, Meng-chang and Ning, 2010, Wong and Kim, 2010). The SMC solutions are categorized in Figure 2.3.

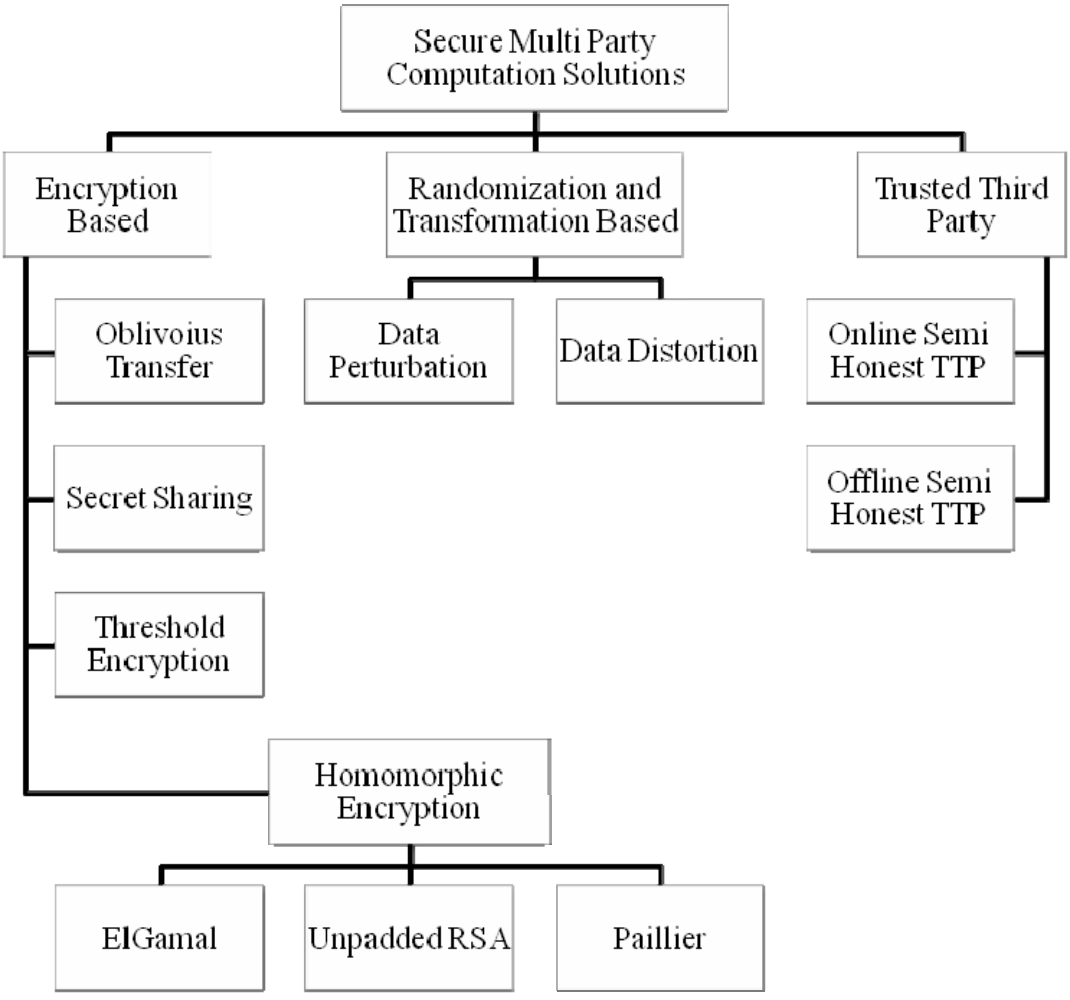


Figure 2.3 Secure multi party computation solutions

The section continues with the description of the secure multi party computation solutions.

2.5 Encryption Based Solution

In this section an overview of encryption based techniques is provided.

2.5.1 Oblivious Transfer

The problem of Oblivious Transfer was introduced by Rabin (Rabin, 1981). Goldreich *et al*. (Even *et al.*, 1985) modeled OT in a two party system in which one party, say *A*, has a bit *i* and the other party, say *B*, has two bits b_0 and b_1. *A* wants to get b_i from *B* without letting him know *i*. 1-Out-Of- 2 OT was generalized to 1-OUT-OF-*n* OT by Brassard *et al*. (Brassard *et al.*, 1987), in which *B* has an array of size *n* and *A* wants to get the *i*th element. A variant of oblivious transfer entitled 1-out-of-2 oblivious transfer protocol was developed from Rabin's original model (Ishai *et al.*, 2008, Pinto *et al.*, 2009, Schaffner, 2010).

In term of the privacy of 1-out-of-2 oblivious transfer (Jain and Hari, 2010, Vanishree and Iyengar, 2009), suppose that the sender blindly sends the recipient two secret messages, m_1 and m_2. If the sender executes the protocol correctly then the recipient can read exactly one of the messages, m_1 or m_2. The probability that the recipient has received message m_1 is 0.5 and therefore the sender cannot tell which message the recipient has read.

2.5.2 Secret Sharing

Secret sharing is the method of sharing a secret by multiple parties, so that no one and no party know the secret, but the secret could be constructed by combing some parties' shares. A multi-secret sharing scheme is a protocol to share a number of (arbitrarily related) secrets among a set of participants in such a way that only qualified sets of participants can recover the secrets (Blundo and Masucci, 1999).

For example, in a two-party case, Alice and Bob share a value x modulo some appropriate value N, in such a way that Alice holds a, Bob holds b, and x is equal to $(a+b)$ *mod* N. This is called additive secret sharing. An important property of this kind of secret sharing is that if Alice and Bob have shares of a and b, then they can each locally add their shares modulo N to obtain shares of $a+b$.

Shamir secret sharing is a threshold scheme (Shamir, 1979). In Shamir secret sharing, there are N parties and a polynomial P of degree k-1 such that $P(0)=x$ where x is a secret. Each of the N parties holds a point in the polynomial P. Because k points (x_i, y_i) $(1 \leq i \leq k)$ uniquely define a polynomial P of degree k-1, a subset of at least k parties can reconstruct the secret x. But, fewer than k parties cannot construct the secret x. This scheme is also called (N, k) *Shamir secret sharing*.

The shared secret usually computes from a combination of all participants' public keys, so all parties should present their portion of the shared secret to create a shared key. The process of gathering all participants' portion of shared secret key tends to the offline secret sharing. The offline secret sharing process is displayed in Figure 2.4.

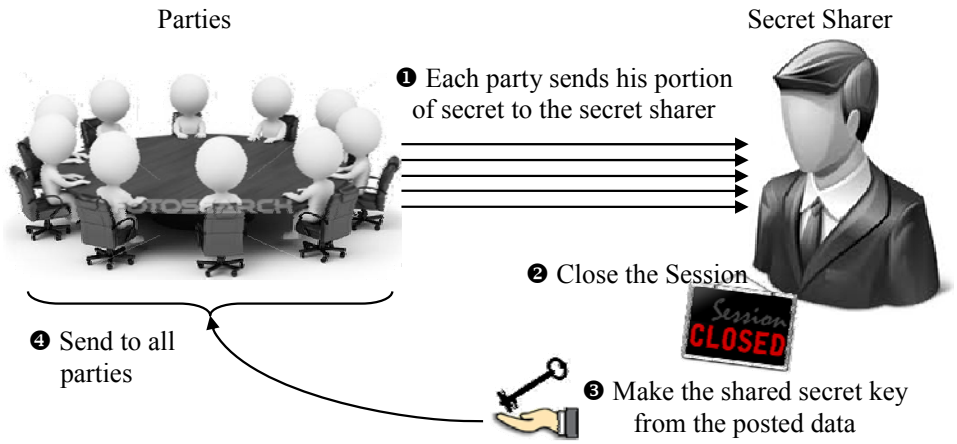


Figure 2.4 Offline Secret Sharing

When the secret shares offline, then the multi party computation executes offline due to the fundamental role of the secret sharing in multi party computation process. The offline secret sharing includes four stages;

1. Each party send his portion of the final shared secret key, usually his public key

2. When all parties send their portion of data, the secret sharer closes the session.

3. Secret sharer makes the shares secret key with combination of the posted data.

4. Secret sharer sends the shared secret key to all participants to enable them for joining the multi party computation process.

There are two types of secret sharers; in the first manner, the secret sharer is a trusted third party and in the second way, each party computes the shared secret himself. Despite the type of the secret sharer, the dependence between the shared secret and the participants, leads to an offline secret sharing process (Farras and Padro, 2009, Li *et al.*, 2010, Luo *et al.*, 2011).

2.5.3 Threshold Encryption

If a private key k is shared among n parties with threshold t, Desmedt and Frankel (Desmedt and Frankel, 1990, Desmedt and Frankel, 1994, Lu and Desmedt, 2010) show that any sub set of t parties can jointly decrypt the ElGamal ciphertexts without explicitly reconstructing k.

Let k_i be the share of the ith party, and $K_i = g^k{}_i$ the commitment to k_i. A sub set T of t parties can decrypt a ciphertext (C_1, C_2) as follows;

$$D_k(C_1, C_2) = \frac{C_1}{{C_2}^k} = \frac{C_1}{\prod_{i \in T}\left(c_2^{k_i}\right)^{\prod_{j \in T, j \neq i} \frac{-j}{i-j}}} \qquad (2.1)$$

It is observed that this equation requires each party $i \in T$ to raise C_2 to the k_i-th power. If only semi-honest adversaries are considered, the proof is not necessary.

2.5.4 Homomorphic Encryption

The main idea of the encryption based secure multi party computation is the homomorphic cryptosystems. Homomorphic cryptosystem is a type of encryption that allows a user to perform an algebraic calculation such as multiplication on the ciphertext and the same algebraic calculation would also be applied to the plain text. Therefore if the ciphertext was multiplied by 2 then when the ciphertext was decrypted, the plain text would also have been multiplied by 2. However fully Homomorphic encryption (Gentry, 2009, Golle and Juels, 2008) that allows both addition and multiplication on ciphertext, is currently too slow for practical use. Inversely some algebraic computation, like addition and multiplication, can be done on original message by doing some algebraic computation on the cipher text, without decrypting the cipher text. In the general manner homomorphic encryption feature can be formalized as;

$$E(msg_1 \times msg_2) = E(msg_1) \times E(msg_2) \qquad (2.2)$$

Where the msg_1 and msg_2 are two different plaintext and E denotes the homomorphic encryption function.

2.5.5 ElGamal Cryptosystem

The ElGamal cryptosystem (ElGamal, 1985) is a part of public encryption systems. The public key is (h, G, q, g) where G is a cyclic group of order q with the generator g, $h=g^x$ and x is the private key which is randomly chosen from [1, q]. All computation in the ElGamal scheme is done in the group G.

Under the public key (h, G, q, g), the ciphertext of a message m (which is the representation of an element of G) is encrypted as

$$E(m) = (c_1, c_2) \; where \;\; c_1 = m.h^r \,, \;\; c_2 = g^r \qquad (2.3)$$

and r is randomly chosen from [1, q]. To decrypt the ciphertext (c_1, c_2) with the private key x, the plaintext message m can be decrypted as $m=c_1(c_2^x)^{-1}$. It clearly is true because

$$c_1(c_2^x)^{-1} = m.h^r(g^{rx})^{-1} = m.h^r(h^r)^{-1} = m \qquad (2.4)$$

ElGamal encryption is semantically secure under the Decisional Diffie-Hellman (DDH) assumption (Boneh, 1998). One family in which DDH is believed to be intractable is the quadratic residue subgroup Q_p of Z_p* where p, q are two primes and $p=2q+1$.

In the ElGamal encryption scheme, one cleartext has many possible ciphertexts because of the random value r. ElGamal supports rerandomization: a new ciphertext $E'(m)$ of m can be computed from a ciphertext $E(m)=(c_1, c_2)$ as $E'(m)=(c_1.h^{r'}, c_2.g^{r'})$ where r' is randomly chosen from [1, q] (Yang, 2007).

The homomorphic property of the ElGamal cryptosystem is proved below;

$$E(m_1) \times E(m_1) = (m_1.h^{r_1}, g^{r_1}) \times (m_2.h^{r_1}, g^{r_1}) = (m_1.m_2.h^{r_1+r_2}, g^{r_1+r_2}) = E(m_1 \times m_2) \qquad (2.5)$$

2.5.6 Unpadded RSA

For description of RSA (Rivest *et al.*, 1978), suppose that p, q are two distinct prime numbers. Let $n = p \times q$ and

$$\varphi(n) = (p-1)(q-1) \qquad (2.6)$$

be the Euler tutient function. Choose an e such that

$$1 < e < \varphi(n) \text{ and } gcd(e, \varphi(n)) = 1 \qquad (2.7)$$

The selected e is the public key and $d = e^{-1} \bmod n$ is the private key. With the plaintext m, the ciphertext is encrypted and decrypted as

$$c = m^e \bmod n, m = c^d \bmod n \quad (2.8)$$

The proof is based on the Euler theorem, Fermat's Little theorem and Chinese reminder theorem as below;

$$c^d \bmod n = (m^e)^d = m^{ed} = m^{1+k\varphi(n)} = m.(m^{\varphi(n)})^k = m \bmod n \quad (2.9)$$

Suppose that m_1, m_2 are two distinct messages, so the homomorphic property of the RSA is proved below;

$$E(m_1) \times E(m_2) = (m_1^e \bmod n) \times (m_2^e \bmod n) = (m_1^e \times m_2^e \bmod n) = ((m_1 m_2)^e \bmod n) = E(m_1 \times m_1) \quad (2.10)$$

2.5.7 Paillier Cryptosystem

The Paillier cryptosystem (Paillier, 1999) is a new public key cryptography introduces in 1999. Suppose that p, q are enough large prime numbers and

$$gcd\,(pq, (p-1)(q-1)) = 1 \quad (2.11)$$

Let $n = pq$ and g is a random number belongs to $\mathbb{Z}_{n^2}$. Compute

$$\lambda = lcm(p-1, q-1) \quad (2.12)$$

$$\mu = L\left(g^\lambda \bmod n^2\right)^{-1} \bmod n \quad (2.13)$$

$$L(u) = \frac{u-1}{n} \quad (2.14)$$

The public key is (n, g) and the private key is (λ, μ). The plaintext m is

encrypted and decrypted with a random number *r* as;

$$c = g^m . r^n \ mod\ n^2 \ , m = L\left(c^\lambda \ mod\ n^2\right) . \mu \ mod\ n \qquad (2.15)$$

The homomorphic addition property of the Paillier cryptosystem is proved below;

$$E(m_1 \times m_2) = g^{m_1} . r_1^n . g^{m_2} . r_2^n \ mod\ n^2 = g^{(m_1+m_2)} (r_1 r_2)^n = E(m_1+m_2) \qquad (2.16)$$

2.5.8 Encryption Based Solution Advantages and Disadvantages

The advantages of encryption-based techniques are that they provide better privacy protection than data distortion techniques and usually require less communication overhead. However the disadvantage of encryption-based techniques is that efficiency of the operation is slow in comparison to data distortion techniques.

2.6 Randomization and Transformation Based Solutions

The random values are usually drawn from either a normal or uniform distribution such that they match the distribution of the original dataset. This allows the newly disguised dataset to keep the near same aggregate statistical data as the original dataset such as a normal distribution, while changing the individual values of the dataset to ensure individual privacy. This allows for meaningful computations to still be carried out.

2.6.1 Data Perturbation

Data Perturbation refers to securing data by adding some randomness to it.

The privacy of a number $t \in T$ is preserved by $t + r$, if the random number r is uniformly distributed in $\left(\frac{-T}{2}, \frac{T}{2}\right)$. In an infinite domain, the generation of random number is still unknown which will lead to some information leakage if the random number r is generated in a finite domain. In a finite field, however, there is no such problem. Data Perturbation causes problems if some operations, such as comparison, are done on the perturbed data. This problem doesn't come if the size of input and random number is less than half the field size, which is a safe assumption for practical purposes.

Data randomization (Agrawal and Agrawal, 2001, Evfimievski *et al.*, 2002, Vaidya and Clifton, 2004) is used to disguise the original data items in a dataset by either adding a random value, r_n, to the data item, x_n, (Agrawal *et al.*, 1993) or by adding or removing random data items to the entire dataset (Kantarcioglu and Clifton, 2002), such that it is difficult to determine which are the original data items (Chen and Liu, 2010, Keke, 2009).

2.6.2 Data Distortion

Due to the low computational cost of data distortion techniques, it is fast becoming a popular approach to SMC. Also for many applications the disclosure of some aggregate statistical information is not a large issue, far out weighted by the benefits of better efficiency. All data distortion techniques attempt hide the data either by the changing of individual values or losing the values in a mass of fake values.

Data swapping (Estivill-Castro and Brankovic, 1999) attempts to hides the relationship between an individual data item, x_i, and the data subject, P_i, to which it belongs. This achieved by creating a new order for the dataset. This new order can then be used to rearrange the original dataset and thereby breaking the link between the data subject and the data item. On its own data swapping cannot preserve the individual data privacy but can preserve the privacy of the link between the data

subject and the data item. However the protection offered by data swapping alone is weak as a party with access to multiple items from the dataset may be able to restore the link between the data subject and the data item or derive the aggregate statistical information of the dataset with some degree of confidence.

Data transformation (Du and Zhan, 2002, Kabir *et al.*, 2007 , Peng *et al.*, 2010) changes the original format of the original dataset to a new format that allows for a computation to take place while preserving the privacy of the original dataset. A linear transformation disguise is an example of data transformation used in (Du and Zhan, 2002) to create the "Two Party Scalar Protocol". It uses an invertible $n \times n$ matrix, M, to transform the two original datasets, X and Y, via matrix multiplication. This creates two matrices, X' and Y', such that the partial disclosure of either matrix does not allow the derivation of the original datasets. This allows both parties to share half of their respective datasets without revealing any identifiable data and perform the scalar product XY. This is secure as long as X and Y are in real number domain and the number of elements in X and Y is not small.

A polynomial function disguise (Du and Zhan, 2002) can be used to hide a secret dataset $Z = \{z_1, z_2, \ldots, z_n\}$ by using a polynomial function of degree k. Each z_i is used in the formula

$$f(x) = a_k x^k + a_{k-1} x^{k-1} + \cdots + a_1 x + a_0 \qquad (2.17)$$

to compute $f(z_i)$. The holder of $f(z_i)$ does not know the set of parameters $A = \{a_0, a_1, \ldots, a_k\}$ and therefore cannot find z_i without also finding $k + 1$ parameters. An additional benefit of this approach is that $f(z_i) > f(z_j)$ if $z_i > z_j$ and all values in Z are positive. This allows a party with access only to $f(z_1), \ldots, f(z_n)$ but not all of Z to rank them in order and find the maximum and minimum values. This could be used to easily solve the millionaire's problem in Yao's original paper (Yao, 1982). The security of the polynomial disguise function is based upon the degree k. By increasing k the security of the function can increase, however decreasing k increases the efficiency of the function at the cost of security.

2.6.3 Randomization Based Solution Advantages and Disadvantages

The advantages of data distortion techniques are their improved efficiency over encrypted-based techniques. However due to the fact that data distortion techniques change either individual data items or the distribution data for a given dataset, this results in a lower accuracy in computation of the function, F, than when using encryption-based techniques.

2.7 Trusted Third Party

The simplest and most general approach to solve the secure multi party computation is to use the Trusted Third Party (TTP) atop of all participants which perform all joint computations and maintain security. Therefore, no party P_i can find out input from other organizations P_j, where $i \neq j$.

The trusted third party was introduced first in (Du *et al.*, 2004, Du and Zhan, 2002, Goldreich, 1998, Goldreich, 2005) to perform the secure multi party computation. All parties provide their inputs, d_i, to the third party and it returns the result, $f(d_1,\ldots,d_n)$, to all participants. If the third party colludes with either party then the privacy of the original input data is compromised. If the third party itself is compromised then the privacy of input and output data for all parties would also be compromised. Additionally neither party may be able to find a third party that they can trust or one that all can trust.

The third party protocol is appealing due to the low communication cost and efficiency, but due to the privacy limitations conclude that in the current business environment the risk is too high to rely on an untrusted third party (Hong *et al.*, 2010, Hong *et al.*, 2009).

Since it can be difficult to find a trusted third party, a semi-trusted third party can instead use to perform either the computation on encrypted or disguised input

data as in online STTP, or to provide noise data to disguise the input data before it is sent to the other party to perform intermediate computations as in offline STTP.

2.7.1 Online Semi-trusted Third Party (STTP)

The online STTP approach means that the STTP is still trusted to perform some of the computations. However the STTP's access to the input data is usually restricted by either encryption or data distortion.

An online STTP is used in (Cachin, 1999) to blindly compare the bits of the two inputs, x and y, supplied by the parties. Once a point is found where the bits differ then the result is registered that will determine whether $x > y$. The third party does not have access to the result. This result is then retrieved by the two parties using Private Information Retrieval (PIR) (Rebollo-Monedero and Forne, 2010).

In (Franklin and Reiter, 1997) the n parties, $P_1,\ldots,P_n$, want to exchange keys, $K_1,\ldots,K_n$ respectively, via a STTP without disclosing the keys to the STTP, or to the other party if the other party tries to be dishonest by not supply the correct key. The solution proposed is that K_i is shared between all P_j ($j \neq i$) and the STTP using a n-out-of-n verifiable secret sharing scheme (Chor *et al.*, 1985, Even *et al.*, 1985). This ensures privacy as only part of the key is shared to the other party and the other part of the key to the STTP; however the original key cannot be derived from one of the parts. The STTP is able to verify that the part of the key it has is correct. Once the STTP verifies that its key parts are correct and the parties have swapped the other parts of the key fairly then it will release its part of the key to both parties. Now all parties have all parts of the other key, they can combine then to form the entire key.

The advantage of both of these schemes is that they are efficient in communication; very few messages have to be sent. They are also secure even if one party decides to be malicious. However the disadvantages are that they are requiring computationally expensive encryption schemes and Franklin (Franklin and Reiter, 1997) also acknowledges that their solution provides little privacy protection when

two of the parties conspire against any other party (Shao, 2010, Wong and Kim, 2010).

2.7.2 Offline Semi-trusted Third Party (STTP)

The offline STTP approach (Du *et al.*, 2004) uses a Commodity Server model, first proposed by Beaver (Beaver, 1997) in 1997. The commodity server is a third party that takes aggregate statistical information such as the mean and variance of the data sets that will be used. It then uses these values to compute random noise called commodities, which then can be added to the data sets to distort the original data. These distorted datasets are passed to the second party where the computation will be done. The third party does no computation itself and has no access to any original datasets (Hussein and AlMukhtar, 2009).

2.7.3 Trusted Third Party Advantage and Disadvantages

The strengths of this approach over multi party computation as pointed out in (Du *et al.*, 2004) are that the third party cannot derive the original datasets and cannot view the intermediate results. The third party will also follow the protocol correctly and not collude with either party. Additionally it usually provides a more efficient solution than the two-party solution.

The disadvantage is that the parties are still required to disclose some aggregate statistical data to the third party (Du *et al.*, 2004, Ma and Sivakumar, 2006). Additionally, the second party may still be able to compute an approximation of the aggregate statistical data from the distorted dataset.

2.8 Multi Party Application

Large amount of work has been done on secure multi party computation to provide secure joint computations among mutually distrusted entities. This computation can be anything like selective information sharing, arithmetic/relational operations, sorting, searching, hashing or other such operations. Goldreich (Goldreich, 1998, Goldreich, 2008) is prominent researcher who contributed a lot to SMC in the form of secure solutions for any functionality. Besides this, Agrawal *et al*. (Agrawal and Srikant, 2007) provided fast and secure algorithms for mining association rules; Atallah *et al*. (Du and Atallah, 2001) contributed to secure multiparty computation geometry, which are a base for routing and other network related problems. Lindall *et al*. (Lindell and Pinkas, 2007) provided cryptographic techniques and solutions for secure multi party computation. Rebecca Wright (Wright, 2008) provided some solutions to SMC and Privacy Preserving data mining through its PORTIA project. Several problems and protocols to solve them have also been proposed by various eminent researchers which provide a clear view of SMC, their problems and solutions. Some examples are Privacy Preserving Data Mining (Agrawal and Srikant, 2000, Keke, 2009, Peng *et al.*, 2010), Privacy Preserving Graph Algorithms in the Semi-honest Model (Brickell and Shmatikov, 2005), Privacy Preserving Cooperative Scientific Computation (Atallah and Du, 2001), Privacy Preserving Set Union (Frikken, 2007), Privacy Preserving Electronic Surveillance (Frikken and Atallah, 2003), Privacy Preserving Cooperative Statistical Analysis (Du and Atallah, 2002). One of the most interested areas is Privacy Preserving Computational Geometry (PPCG). Privacy Preserving Computational Geometry aims to solve problems of Computational Geometry in a secure manner. Some of works done in this area are Privacy Preserving Shortest Path Computation (Das *et al.*, 2008), Secure multi-party computation problems and their applications (Du and Atallah, 2001), Secure Multi-party Computational Geometry (Atallah *et al.*, 2001, Chen and Liu, 2010) and Secure Two-party Protocols for Point Inclusion Problem (Meng-chang and Ning, 2010, Nielsen and Orlandi, 2009, Pinkas *et al.*, 2009, Yao, 1986).

There are a lot of application could be solved by secure multi party

computation algorithms which some of them are listed below;

- Let *n* research universities from various countries wish to discover some current research trends from their research databases without compromising the security of each individual database.
- Let all universities across the globe jointly wish to evaluate each other and then declare the top 20 universities of the world on the basis of their 5 year's academic records. They all want to preserve the privacy of their individual databases.
- Let us consider that several shopkeepers of some general stores wish to find shopping trends of customers/buying patterns without revealing information about their databases.
- Let us consider an Intelligence Department that considers database of fingerprints/thumb impressions. Now, if some police station employee wishes to check a particular fingerprints, it must not be able to gain its complete access, instead, he should only get the test results.
- Alternatively, if police wishes to check a particular person's identity from his thumb impression/signature, they can consult the bank database. Bank database only reveals the match results of thumb impression/signature.
- Let us consider *n* hospitals situated in various different countries having their medical databases and patient's history stored on some remote database sites. If an insurance company wishes to verify the med claim of a particular person, he can get that patient's information from hospital's database, but the hospital's database is not completely provided, instead only the requested information is allowed to access.
- Let all doctor's team from several countries want to jointly find a remedy for a particular disease. All carry out research and studies and only reveal conclusions before each other without revealing the whole task.

- Let us consider Airlines Company that has a reservation database for each country. If a person wishes to make a reservation from city *a* located in country *A* to a city *b* located in country *B*, then he need to consult each intermediate countries databases. These databases provide only the queried details without disclosing their whole reservation database.

- Let a social organization providing funds to large number of charitable trusts located in different countries. These charitable trusts can query the organization to check whether the requested fund has been issued or not, but cannot see the organization's whole database.

- Several websites provides ocean of knowledge and contains authentication information. Whenever, someone do e-shopping /e-commerce, the authentication database first validates us as an authenticated user and then when it comes to payment, our account number/credit card number is checked for correctness in the bank database and if transaction successfully completes, then only item is said to be purchased. In this case, authentication only checks the individual person's identity and bank's database check the card number only and other authentication and the bank database is kept confidential.

The section continues with some new SMC problems and their applications along with the solutions.

2.8.1 Privacy Preserving Co-operative Scientific

Let Alice has *m* private linear equations represented as $M_1 \times X = B_1$ and Bob has $(n - m)$ private linear equations represented by $M_2 \times X = B_2$, where X is an n dimensional vector. Alice and Bob wish to jointly find a vector X that could satisfy both Alice and Bob's equations (Atallah and Du, 2001).

Linear Least Square Problem; Let Alice has m_1 private linear equations represented by $M_1 \times X = B_1$ and Bob has m_2 private linear equations represented as

$M_2 \times X = B_2$, where X is an n dimensional vector and $m_1 + m_2 > n$. Since there are more conditions to be satisfied than the degree of freedom, it is possible that some of them may be violated. Therefore, the residual factor r is taken such that r is kept as minimum as possible. The least square criterion is the use of Euclidean (least square) norm for size of r (Du *et al*., 2004).

Linear Programming Problem; Let Alice has private linear system of equations represented as $M_1 \times X \leq B_1$ and Bob has private linear system of equations represented as $M_2 \times X \leq B_2$, where Alice has m_1 linear equations in her system and Bob has m_2 linear equations. The goal is to minimize $A_1 \times X_1 + A_2 \times X_2 + \ldots + A_n \times X_n$ for known $A_1 \ldots A_n$ and the solutions $X = (X_1 \ldots X_n)$ should satisfy both Alice and Bob's requirements. These problems are generally viewed as routing, planning, scheduling, assignment and design (Du *et al.*, 2004).

2.8.2 Privacy Preserving Database Query

Suppose Alice want to search a string q in Bob's database of strings $S = \{S_1, \ldots, S_n\}$ and it just want to return the result, without revealing the Bob's entire string database. The match could be exact or approximate match (Rebollo-Monedero and Forne, 2010).

2.8.3 Privacy Preserving Intrusion Detection

Profile Matching; Alice has a database of hacker's profile. Bob has recently traced a behavior of a person, whom he suspects a hacker. Now, if Bob wants to check whether his doubt is correct, he needs to check Alice's database. Alice's database needs to be protected because it contains hacker's related sensitive information. Therefore, when Bob enters the hacker's behavior and searches the Alice's database, he can't view his whole database, but instead, only gets the comparison results of the matching behavior (Park *et al.*, 2007).

Fraud Detection; Two major financial organizations want to cooperate in preventing fraudulent intrusions into their computing system, without sharing their data patterns, since their individual private database contains sensitive data (Androulaki and Bellovin, 2010).

2.8.4 Privacy Preserving Data Mining

Classification; Alice has a private database DB_1 and Bob has private database DB_2. How can Alice and Bob build a decision tree based on $DB_1 \cup DB_2$ without disclosing the contents of their private database to each other? Several algorithms like ID3, Gain Ratio, Gini Index and many other can be used for Decision Tree along with SMC protocols (Ge *et al.*, 2005).

Data Clustering; Alice has a private database DB_1 and Bob has private database DB_2. Alice and Bob want to jointly perform data clustering on $DB_1 \cup DB_2$. This is primarily based on data clustering principle that tries to increase intraclass similarity and minimize interclass similarity (Shen *et al.*, 2010).

Data Generalization, Summarization and Characterization; Let Alice has a private database DB_1 and Bob has private database DB_2. If they wish to jointly perform data generalization, summarization or characterization on their combined database $DB_1 \cup DB_2$, then this problem becomes an SMC problem (Inan *et al.*, 2007).

Mining Association Rules; Let Alice has a private database DB_1 and Bob has private database DB_2. If Alice and Bob wish to jointly find the association rules from $DB_1 \cup DB_2$ without revealing the information from individual databases. The distributed association rule mining is fully discussed subsequently and is implemented and tested in Chapter 5 to ensure the practicability of designed secure multi party computation algorithm (Zhong and Sheng, 2007).

In case of solving above applications they wish to apply any selection procedure on each other's databases, and then such a process should not reveal their database knowledge to the other party or even they jointly wish to sort their database without disclosing each other's database.

2.8.5 Privacy Preserving Geometric Computation

Intersection; Let Alice has a private shape MAP_a and Bob has private shape MAP_b, if Alice and Bob want to find whether MAP_a and MAP_b intersect, then they need to share their database of shape coordinates to find whether they intersect (Chen and Liu, 2010).

Point Inclusion Problem Let Alice has a private shape MAP_a and Bob has private point p. Now, if Bob wants to know whether his private point p lies on shape boundary or inside or outside, then they need to jointly use both databases without revealing their individual information to each other (Keke, 2009).

Range Searching; Let Alice has a private range and Bob has N private points. Alice and Bob want to jointly find the number of points in the Alice's range; neither is willing to disclose their data to other party (Atallah *et al.*, 2001).

Closest Pair; Let Alice has M private points and Bob has N Private points in a plane. Alice and Bob want to jointly find the two points closest among (M+N) points, for example two points having their mutual distance smallest. They may wish to find the shortest path among the two locations a and b (Keke, 2009).

2.8.6 Privacy Preserving Statistical Analysis:

Correlation and Regression; Let $D_1=(X_1,\ldots, X_n)$ be Alice's private dataset and $D_2=(Y_1,\ldots, Y_n)$ be Bob's private dataset. Alice and Bob wish to jointly find the

Correlation Coefficient between X and Y between the private datasets D_1 and D_2 without revealing D_1 or D_2 to each other or find the regression lines for D_1 and D_2 and perform regression analysis for future predictions (Jing *et al.*, 2007).

2.8.7 E-Voting

Voting schemes are one of the most important examples of advanced cryptographic protocols with immediate potential for practical applications. The most important goals for such protocols are

- **Privacy**: only the final result is made public, no additional information about votes will leak.
- **Robustness**: the result correctly reflects all submitted and well-formed ballots, even if some voters and/or possibly some of the entities running the election cheat.
- **Universal Verifiability**: after the election, the result can be verified by anyone.

In (Fouque *et al.*, 2001) voting was pointed out as a potential application for their cryptosystem. However, no suggestion was made for protocols to prove that an encrypted vote is correctly formed, something that is of course necessary for a secure election in practice (Blosser and Zhan, 2008, Li *et al.*, 2009 , Otsuka and Imai, 2010). In work done concurrently with and independent from ours, Stern, Baudron, Fouque, Pointcheval and Poupard (Baudron *et al.*, 2001) propose a voting scheme somewhat similar to ours. Their work can be seen as being complementary to ours in the sense, that their proposal is more oriented toward the system architectural aspects of a large scale election, and less toward optimization of the building blocks. To compare with their scheme, note that their modulus length k must be chosen such that $k > L \cdot \log(M)$. The scheme produces ballots of size $O(k \cdot L)$. An estimate with explicit constants is given in (Baudron *et al.*, 2001), in which the dominating term in our notation is $9kL$. Because our voting scheme uses the generalized Paillier

cryptosystem, k can be chosen independently from L, M. In particular the scheme can scale to any size of election, even after the keys have been generated. However, if k chooses as in (Baudron *et al.*, 2001), i.e. $k > L \cdot log(M)$, then the produced ballots have size $O(k \cdot log(L))$. Working out the concrete constants involved, one finds that our complexity is dominated by the term $10k\ log(L)$. So already for moderate size elections they have gained a significant factor in complexity compared to (Baudron *et al.*, 2001, Furukawa *et al.*, 2009). Hirt and Sako (Hirt and Sako, 2000) propose a general method for building receipt-free election schemes, i.e. protocols where vote-buying or -coercing is not possible, because voters cannot prove to others how they voted. Their method can be applied to make a receipt-free version of the scheme from (Cramer *et al.*, 1997). Damg˚ard and Koprowski (Damg˚ard and Koprowski, 2001, Damgård and Thorbek, 2007) propose techniques by which one can drop these restrictions from Shoup's scheme at the expense of an extra intractability assumption.

2.9 Secure Distributed Frequent Itemset Mining

Data mining tools deals with a huge volume of data which can find a small piece of gold hidden in a corner. Return costs spent on these tools is often very rapid. Data mining tools searches the patterns and grouped data that may remain hidden from our view. The data mining tool assumes that you just do not know what you want.

Data mining are classified in multi variant fields and applications, but this research only focuses on distributed frequent itemset mining over vertically partitioned databases. The data mining domain and the research goal are shown in Figure 2.5.

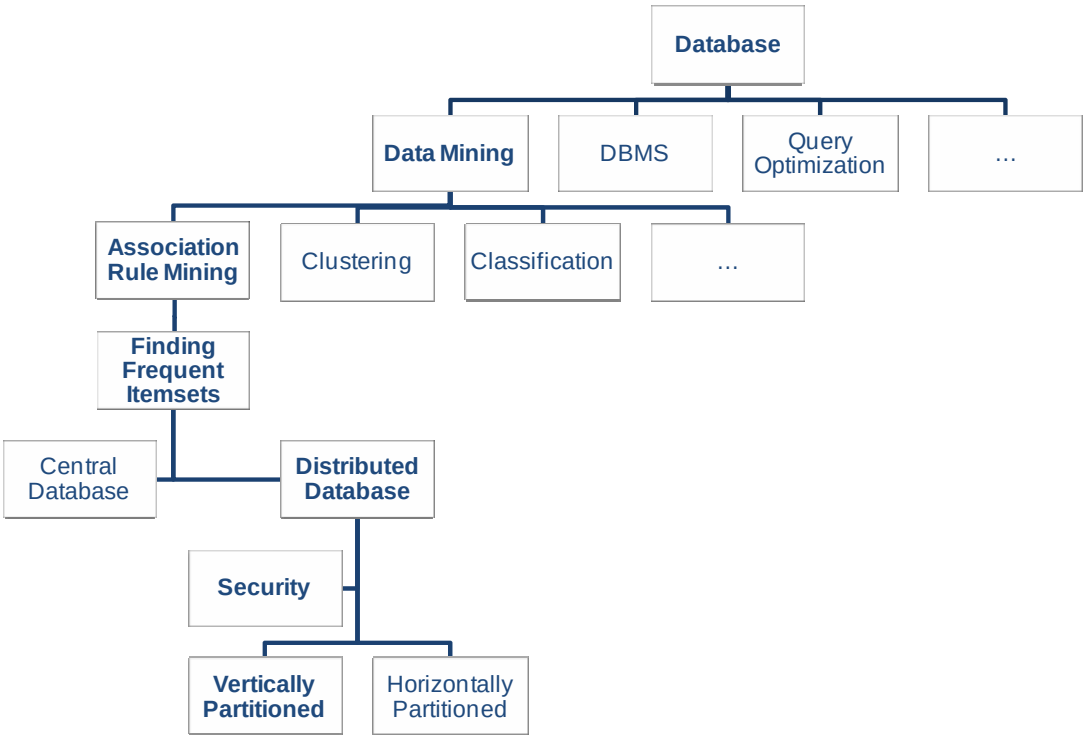


Figure 2.5 Data mining domain; the research focus is bolded

2.10 Privacy Preserving Frequent Itemset Mining

To the best of our knowledge, Clifton and his students were the first to study privacy preserving distributed mining of association rules and frequent itemsets. In (Wright and Yang, 2004) Vaidya and Clifton gave a nice algebraic solution for vertically partitioned data. However, this solution can leak many linear combinations of each party's private data to the other. Furthermore, to process one candidate frequent itemset, its computational overhead is quadratic in the number of transactions. In (Kantarcioglu and Clifton, 2002) Kantarcioglu and Clifton gave a solution for horizontally partitioned data that uses Yao's generic secure computation protocol as a sub protocol. However, as Goldreich pointed out in (Goldreich, 1998), generic secure computation protocols are highly expensive for practical purposes. (In data mining problems, because the input size is huge, they can be even more expensive than in other applications.) Furthermore, the solution in (Kantarcioglu and Clifton, 2002) only works for three parties or more, not for two parties.

Privacy-preserving data mining has been a topic of active study (see, e.g., papers by Agrawal et al. (Agrawal and Agrawal, 2001, Agrawal and Srikant, 2000)). In particular, many papers have addressed the privacy issues in mining of association rules and frequent itemsets. Some examples are (Bertino *et al.*, 2005, Chiew, 2008, Dasseni *et al.*, 2001, Evfimievski *et al.*, 2002, Peng *et al.*, 2005, Rizvi and Haritsa, 2002, Saygin *et al.*, 2002, Sherman *et al.*, 2009, Stanley *et al.*, 2002, Urabe *et al.*, 2007, Yu *et al.*, 2006, Zhong and Sheng, 2007). However, these papers are concerned with privacy of individual transactions and/or hiding of sensitive rules (Yu *et al.*, 2006, Zhong and Sheng, 2007).

Privacy preserving distributed mining was first addressed by (Lindell and Pinkas, 2000), but their paper only discusses the classification problem (‘‘classifying transactions into a discrete set of categories’’), not the association rule problem.

As pointed out by Du and Atallah (Du and Atallah, 2001), the problems of privacy preserving data mining can be viewed as an application of generic secure computation. Existing protocols for generic secure computation (Chaum *et al.*, 1988, Damgård and Jurik, 2001, Goldreich *et al.*, 1991) can solve such problems in theory. However, these generic protocols are highly expensive; and therefore it is our goal to design special purpose solutions that are much more efficient for certain problems of interest.

Goethals *et al*. (Goethals *et al.*, 2005) proposed a solution of privacy preserving algorithm for vertically partition data. Their solution is based on additively homomorphic encryption schemes. The state-of-art additively homomorphic encryption schemes are significantly slower than their multiplicatively homomorphic counter partners. For example, the Paillier encryption scheme (which is a suggested choice of encryption scheme in (Goethals *et al.*, 2005) is about eight times slower than the ElGamal encryption scheme (which is a suggested choice of encryption scheme in this research) in both encryption and decryption.

The improved versions by (Damgård and Jurik, 2001, Damgård and Thorbek, 2007) have different efficiency for different operations, but they are all significantly

slower than ElGamal. Consequently, our solution is significantly more efficient than Goethals, *et al*.'s protocol. Assuming the Paillier encryption scheme is used in (Goethals *et al.*, 2005), their protocol is about three times faster than (Goethals *et al.*, 2005).

(Wright, 2008, Wright and Yang, 2004) gave another protocol based on additively homomorphic encryption, which could be used to solve our problem on vertically partitioned data. Assuming that they use Paillier encryption scheme as suggested in their paper.

(Rozenberg and Gudes, 2003) also worked on this problem and gave practical solutions for vertically partitioned data. But as they mentioned in (Rozenberg and Gudes, 2003), their solutions are not ''zero-knowledge'' (i.e., not fully private in the cryptographic sense).

2.10.1 Association Rule Mining

Following the original definition by Agrawal et al (Agrawal *et al.*, 1993) the problem of association rule mining is defined as: Let $I=\{i_1,i_2,\ldots,i_n\}$ be a set of n binary attributes called *items*. Let $D=\{t_1,t_2,\ldots,t_m\}$ be a set of transactions called the *database*. Each transaction in D has a unique transaction ID and contains a subset of the items in I. A *rule* is defined as an implication of the form $X \Rightarrow Y$ where $X,Y \subseteq I$ and $X \cap Y = \varnothing$. The sets of items (for short *itemsets*) X and Y are called *antecedent* (left-hand-side or LHS) and *consequent* (right-hand-side or RHS) of the rule.

To illustrate the concepts, a small example from the supermarket domain is used. The set of items is $I=\{milk,bread,butter,beef\}$ and a small database containing the items (1 codes presence and 0 absence of an item in a transaction) is shown in the table 2.1. An example rule for the supermarket could be $\{milk,bread\} \Rightarrow \{butter\}$ meaning that if milk and bread is bought, customers also buy butter.

Table 2.1 Example data base with 4 items and 5 transactions

Transaction ID	Milk	Bread	Butter	Beef
1	1	1	0	0
2	0	1	1	0
3	0	0	0	1
4	1	1	1	0
5	0	1	0	0

To select interesting rules from the set of all possible rules, constraints on various measures of significance and interest can be used. The best known constraints are minimum thresholds on support and confidence. The *support supp*(X) of an itemset X is defined as the proportion of transactions in the data set which contain the itemset. In the example database, the itemset {*milk, bread*} has a support of 2/5=0.4 since it occurs in 40% of all transactions (2 out of 5 transactions).

The confidence of a rule is defined as;

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} \qquad (2.18)$$

For example, the rule {*milk, bread*}⇒{*butter*} has a confidence of 0.2 / 0.4 = 0.5 in the database, which means that for 50% of the transactions containing milk and bread the rule is correct. Confidence can be interpreted as an estimate of the probability $P(Y|X)$, the probability of finding the RHS of the rule in transactions under the condition that these transactions also contain the LHS (Hipp *et al.*, 2005).

The lift of a rule is defined as;

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(Y).supp(X)} \qquad (2.19)$$

or the ratio of the observed confidence to that expected by chance. The rule {*milk, bread*}⇒{*butter*} has a lift of 0.2/(0.4×0.4)=1.25.

The conviction of a rule is defined as;

$$conv(X \Rightarrow Y) = \frac{1-supp(Y)}{1-conf(X \Rightarrow Y)} \quad (2.20)$$

The rule $\{milk, bread\} \Rightarrow \{butter\}$ has a conviction of (1-0.4)/(1-0.5)=1.2, and be interpreted as the ratio of the expected frequency that X occurs without Y if they were independent to the observed frequency.

Association rules are required to satisfy a user-specified minimum support and a user-specified minimum confidence at the same time. To achieve this, association rule generation is a two step process. First, minimum support is applied to find all frequent itemsets in a database. In a second step, these frequent itemsets and the minimum confidence constraint are used to form rules. While the second step is straight forward, the first step needs more attention. Finding all frequent itemsets in a database is difficult since it involves searching all possible itemsets (item combinations). The set of possible itemsets is the power set over I and has size $2^n - 1$ (excluding the empty set which is not a valid itemset). Although the size of the power set grows exponentially in the number of items n in I, efficient search is possible using the downward closure property of support (Agrawal *et al.*, 1993) which guarantees that for a frequent itemset also all its subsets are frequent and thus for an infrequent itemset, all its supersets must be infrequent. Exploiting this property, efficient algorithms like Apriori (Agrawal and Srikant, 2007), Eclat (Zaki, 2006) and FPGrowth (Han *et al.*, 2004) can find all frequent itemsets.

2.10.2 Apriori Algorithm

Most of the frequent itemset algorithms are slight modifications of the Apriori (Agrawal *et al.*, 1993) algorithm. The Apriori uses explicitly anti-monotone relation between itemsets and their supports, formally described by the rule

$$X \subseteq Y \Rightarrow supp(X) \geq supp(Y) \quad (2.21)$$

The rule follows immediately, since all records containing items *Y* contain *X* as well. Shortly put, all subsets of the frequent itemset are also frequent. So instead of searching over all possible subsets, the algorithm starts from one element itemsets, and then tests only those two element sets that have frequent subsets and so on. It is easy to see that for candidate *Y* with cardinality *l* it is sufficient to check whether all *l*−1 element subsets are frequent. Algorithm 2.1 is ascetic description of the Apriori Algorithm.

```
Input: Support threshold s
F = ∅; Z = I; l = 1
while |Z| > 0 do
      // Find all valid candidates
      F = F ∪ {X ∈ C : supp(X) > s}
      // Form l-element candidate set
      C = {Y ∈ P(I) : |Y| = l, X ⊆ Y ⇒ X ∈ F}
      l = l + 1
return F
```

Algorithm 2.1 Apriori Algorithm

The size of output is determined by the threshold *s*. If *s* is small then the number of frequent itemsets is large. The working time depends linearly on the size of database and on the output size. In that sense the Apriori algorithm is optimal. Of course, the Apriori will slow down, if there is even a single large frequent set. Therefore, the Apriori is suitable for mining sparse records with no large frequent itemsets.

2.10.3 FP-Growth Algorithm

(Han *et al.*, 2004) proposed the FP-growth algorithm; the first pattern-growth concept algorithm. FP-growth constructs an FP-tree structure and mines frequent

patterns by traversing the constructed FP-tree. The FP-tree structure is an extended prefix-tree structure involving crucial condensed information of frequent patterns.

The FP-tree structure has sufficient information to mine complete frequent patterns. It consists of a prefix-tree of frequent 1-itemset and a frequent-item header table. Each node in the prefix-tree has three fields: *item-name*, *count*, and *node-link*. *Item-name* is the name of the item. *Count* is the number of transactions that consist of the frequent 1-items on the path from root to this node. *Node-link* is the link to the next same item-name node in the FP-tree.

Each entry in the frequent-item header table has two fields: *item-name* and *head of node-link*. *Item-name* is the name of the item. *Head of node-link* is the link to the first same item-name node in the prefix-tree.

FP-growth has to scan the *TDB* twice to construct an FP-tree. The first scan of *TDB* retrieves a set of frequent items from the *TDB*. Then, the retrieved frequent items are ordered by descending order of their supports. The ordered list is called an F-list. In the second scan, a tree *T* whose root node *R* labeled with "null" is created. Then, the following steps are applied to every transaction in the *TDB*. Here, let a transaction represent [*p* | *P*] where *p* is the first item of the transaction and *P* is the remaining items.

In each transaction, infrequent items are discarded. Then, only the frequent items are sorted by the same order of F-list. Call insert_tree (*p* | *P*, *R*) to construct an FP-tree. The function insert_tree (*p* | *P*, *R*) appends a transaction [*p* | *P*] to the root node *R* of the tree *T*. Pseudo code of the function insert_tree (*p* | *P*, *R*) is shown in Algorithm 2.2.

```
function insert_tree ( p | P, R) {
      let N be a direct child node of R, such that N 's
      item-name = p 's item-name.
      if ( R has a direct child node N )
            increment N 's count by 1.
```

```
else
        create a new node M linked under the R .
        set M 's item-name equal to p .
        set M 's count equal to 1.
call insert_tree (P, N) .
```

Algorithm 2.2 Pseudo code of insert_tree (p | P, R)

An example of an FP-tree is shown in Figure 2.6. This FP-tree is constructed from the *TDB* shown in Table 2.2 with *min_sup* = 3. In Figure 2.6, every node is represented by (*item–name*: *count*). Links to next same item-name node are represented by dotted arrows.

Table 2.2 Sample Transactions Database

Trans ID	Items	Frequent Items
1	*f, a, c, d, g, i, m, p*	*f, c, a, m, p*
2	*a, b, c, f, l, m, o*	*f, c, a, b, m*
3	*b, f, h, j, o*	*f, b*
4	*b, c, k, s, p*	*c, b, p*
5	*a, f, c, e, l , p, m, n*	*f, c, a, m, p*

FP-growth mines frequent patterns from an FP-tree. To generate complete frequent patterns, FP-growth traverses all the node-links from "head of node-links" in the FPtree's header table. For any frequent item a_i , all possible frequent patterns including a_i can be mined by following a_i's node-link starting from a_i's head in the FP-tree header table. In detail, a_i's prefix path from a_i's node to root node is extracted at first. Then, the prefix path is transformed into a_i's conditional pattern base, which is a list of items that occur before a_i with the support values of all the items along the list. Then, FP-growth constructs a_i's conditional FP-tree containing only the paths in a_i's conditional pattern base. It then mines all the frequent patterns including item a_i from a_i's conditional FP-tree. For example, Figure 2.6 describes how to mine all the frequent patterns including item *p* from the FP-tree. For node *p*,

FP-growth mines a frequent pattern (*p:*3) by traversing *p*'s node-links through node (*p:2*) to node (*p:*1). Then, it extracts *p*'s prefix paths; <*f:*4,*c:*3,*a:*3,*m:*2> and <*c:*1,*b:*1>. To study which items appear together with *p*, the transformed path <*f*:2,*c:*2,*a:i*,*m:*2> is extracted from <*f*:4,*c:*3,*a:*3,*m:*2> because the support value of *p* is 2. Similarly, we have <*c:*1,*b:*1>. The set of these paths {(*f*:2,*c:*2,*a*:2,*m*:2),(*c*:1,*b*:1)} is called *p*'s conditional pattern base.

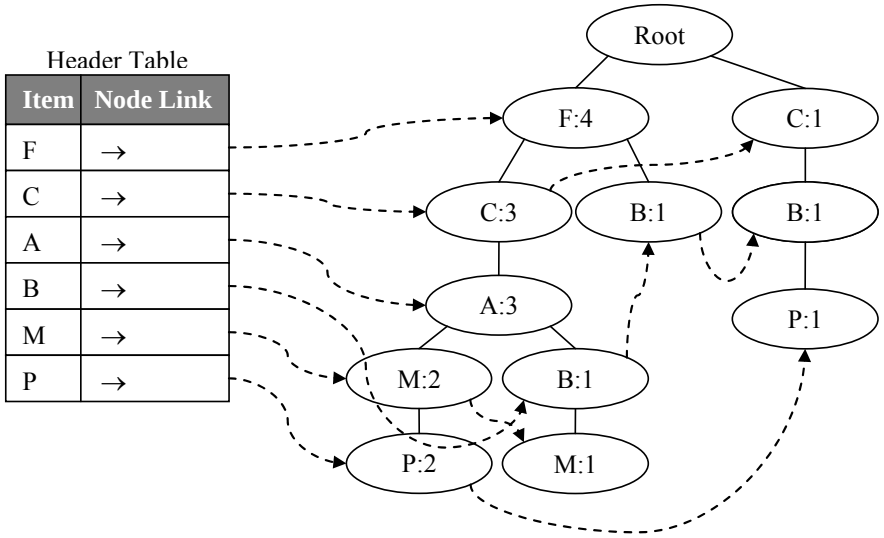


Figure 2.6 Example of an FP-tree

FP-growth then constructs *p*'s conditional FP-tree containing only the paths in *p*'s conditional pattern base. As only *c* is an item occurring more than min_sup appearing in *p*'s conditional pattern base, *p*'s conditional FP-tree leads to only one branch (*c:*3). Hence, only one frequent pattern (*cp:*3) is mined. The final frequent patterns including item *p* are (*p:*3) and (*cp:*3).

2.10.4 Eclat Algorithm

Most frequent itemset mining algorithms as Apriori (Agrawal *et al.*, 1993) and Eclat (Zaki, 2006) use a total order on the items *A* of the alphabet and the itemsets *P(A)* to prevent that the same itemset, called *candidate*, is checked twice for frequency. Items orderings ≤ are in one-to-one correspondence with item codings,

i.e., bijective maps $o:A \rightarrow \{1, \ldots, n\}$ via natural ordering on N. For itemsets $X,Y \in P(A)$ one defines their prefix as

$$prefix(X,Y) = \{\{x \in X | x \leq z\} | maximal\ z \in X \cap Y : \{x \in X | x \leq z\} = \{y \in Y | y \leq z\}\} \quad (2.22)$$

Any order on A uniquely determines a total order on *P(A)*, called lexicographic order, by

$$X < Y \Leftrightarrow \min\left(\frac{X}{prefix(X,Y)}\right) < min\left(\frac{Y}{prefix(X,Y)}\right) \quad (2.23)$$

For an itemset $X \in P(A)$ an itemset $Y \in P(A)$ with $X \subset Y$ and $X < Y$ is called an extension of X. An extension Y of X with $Y = X \cup \{y\}$ (and thus $y > max\ X$) is called an *1-item-extension* of X. The extension relation organizes all itemsets in a tree, called extension tree or search tree.

Eclat starts with the empty prefix and the item transaction incidence matrix $C_\varnothing$, shortly called incidence matrix in the following, and stored sparsely as list of item covers $C_\varnothing = \{(x, T(\{x\})) \mid x \in A\}$. The incidence matrix is filtered to only contain frequent items by

$$freq(C) = \{(x, Tx) \mid (x, Tx) \in C, |Tx| \geq minsup\} \quad (2.24)$$

That represents frequent *1-item-extensions* of the prefix. For any prefix $p \in P(A)$ and incidence matrix C of frequent *1-item-extensions* of p one can compute the incidence matrix C_x of *1-item-extensions* of $p \cup \{x\}$ by intersection rows:

$$Cx = \{(y, Tx \cap Ty) \mid (y, Ty) \in C, y > x\} \quad (2.25)$$

Where $(x, T_x) \in C$ is the row representing $p \cup \{x\}$. C_x has to be filtered to get all frequent *1-item-extensions* of $p \cup \{x\}$ and then this procedure is recursively iterated until the resulting incidence matrix C_x is empty, signaling that there are no

further frequent *1-item-extensions* of the prefix. Exact description of the Eclat algorithm is shown in Algorithm 2.3.

```
Input: alphabet A with ordering ≤,multiset T∈P(A) of
sets of items, minimum support value minsup ∈ N.
Output: set F of frequent itemsets and support counts.
F = {(∅,|T|)}
C∅ = {(x, T({x})) | x ∈ A}
C∅'= freq(C∅)= {(x, Tx) | (x, Tx) ∈ C∅, |Tx| ≤ minsup}
F = {∅}
addFrequentSupersets(∅,C∅')
**********************************
Function addFrequentSupersets()
Input: frequent itemset p ∈ P(A) called prefix,
incidence matrix C of frequent 1-item-extensions of p
Output: add frequent extensions of p to variable F
for (x, Tx) ∈ C do
      q = p ∪ {x}
      Cq = {(y, Tx ∩ Ty) | (y, Ty) ∈ C, y > x}
      Cq'=freq(Cq) = {(y, Ty) | (y, Ty)∈Cq, |Ty|≤ minsup}
      if Cq' ≠ ∅ then
            addFrequentSupersets(q, Cq')
      F = F ∪ {(q, |Tx|)}
```

Algorithm 2.3 Basic Eclat algorithm

2.10.5 Frequent Itemset Mining Algorithms Advantage an Weaknesses

The goal of this section is to find out the best algorithm of Frequent Itemset

Mining based on the online FIMI repository along with several new datasets for benchmarking purposes. More details exist at the http://fimi.cs.helsinki.fi/. The platform was a Pentium4, 3.2 GHz Processor, with 1GB of memory, using a Western Digital IDE 7200rpms, 200GB, local disk. They also used 14 datasets, with an average of 7 values of minimum support.

Figure 2.7 show the ranking of each algorithm running over all datasets. The weight 0.66 has been spotted for best algorithm and 0.33 for second-best algorithm in each dataset. The total of the weights are shown in Figure 2.6 in three different categories: All Frequent Itemset Mining, Closed Itemset Mining and Maximal Itemset Mining. Figure 2.8 show the average timing for each algorithm running over all datasets in logarithmic scale as listed in Table 2.3.

Table 2.3 Association rule mining algorithm comparison

	All FIM		**Closed FIM**		**Maximal FIM**	
	Hi-Supp	**Low-Supp**	**Hi-Supp**	**Low-Supp**	**Hi-Supp**	**Low-Supp**
Apriori	0.66	1.33	1.33	1	1.66	1.66
FP-Growth	0.66	3	4.33	3.66	5.33	4.66
Eclat	0.66	1	1	0.33	1	1

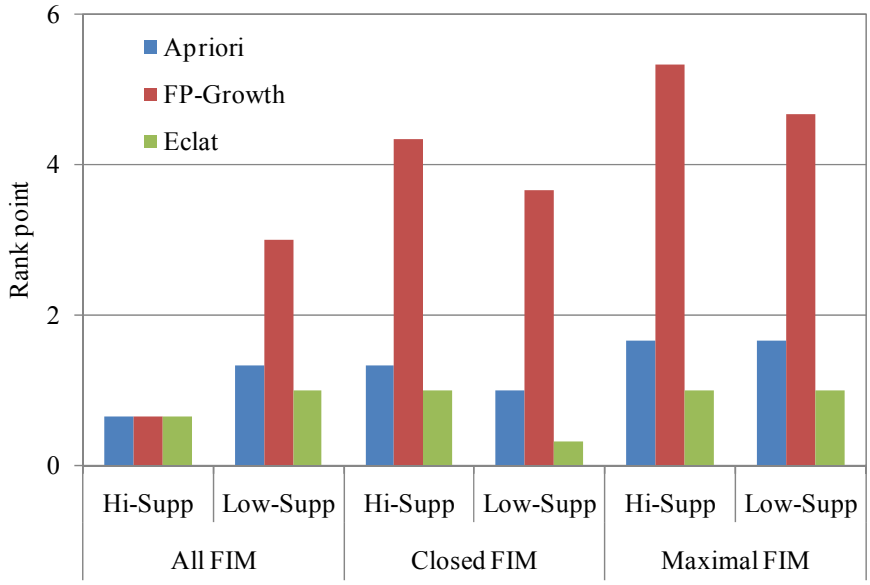


Figure 2.7 Ranking of Each Algorithm Running over All Datasets

All results are not presented in this comparison, because there were a lot of algorithms in FIMI workshop. Base on the FIMI workshop experiments reported, FP-

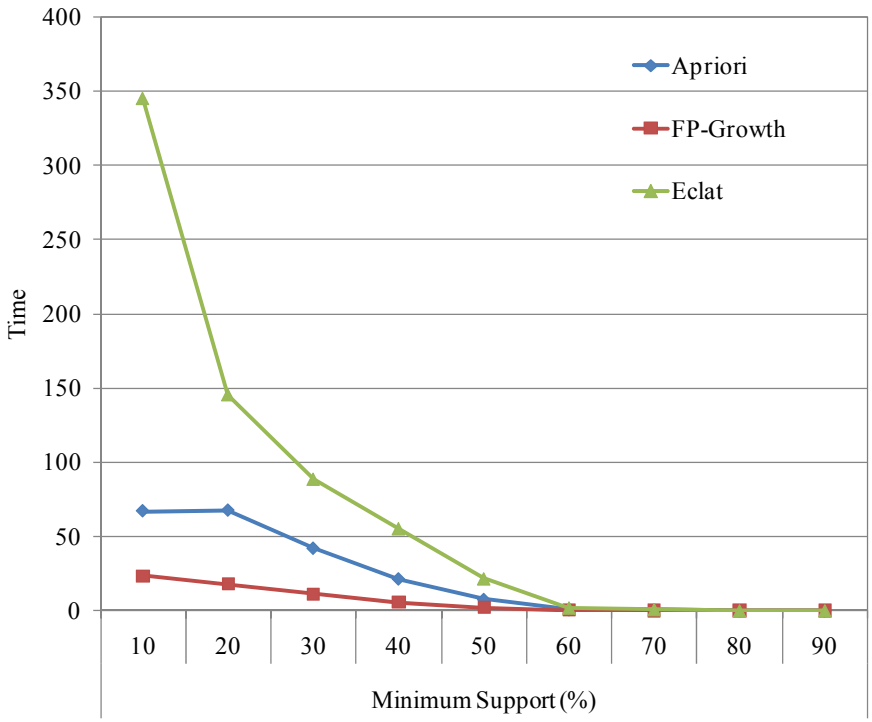


Figure 2.8 Average Timing for Each Algorithm

Growth takes this spot for closed itemset mining, and appears to be one of the best for maximal itemset mining. An interesting observation is that for the synthetic datasets, Apriori seems to perform quite well for all, closed and maximal itemset mining.

2.10.6 Distributed Apriori algorithm

In this section, the unsecured simple distributed Apriori algorithm is explained as the fundamental element of the final algorithm from (Laur, 2004).

Now consider a distributed database shared by t parties $DB=DB_1\cup\ldots\cup DB_t$. Instead of global supports parties can compute local supports $supp_i(A)$. If the number of all records $n = |DB|$ and local records $n_i = |DB_i|$, then the following implication

holds

$$suup(A) > k \Rightarrow \exists i, supp_i(A) > \frac{n_i k}{n} = k_i \quad (2.26)$$

Moreover, there are three classes of frequent itemsets: globally and locally frequent itemsets, and locally supported globally frequent itemsets.

$$F = \{A: suup(A) > k\} \quad (2.27)$$

$$F_i = \{A: suup_i(A) > k_i\} \quad (2.28)$$

$$LF_i = F \cap F_i \quad (2.29)$$

The distributed frequent itemset mining is based on dependencies between F and LF_i. If B be a globally frequent itemset, then there is a site i such that $supp_i(B) > k_i$. Hence, all subsets of B are locally and globally frequent and the implication is;

$$B \in F \Rightarrow \exists i, A \subseteq B \Rightarrow A \in LF_i \quad (2.30)$$

In other words, the Apriori candidate list creates locally. Each party generates candidate sets C_i based on locally supported global sets LF_i, and then tests them locally and afterwards a general candidate list C is formed as a union of new elements in LF_i. Algorithm 2.4 formalizes the idea.

Input: Normalized support threshold k/n.

```
// Calculate local threshold
k_i = n_i k/n
// No frequent sets, candidate set I
F = ∅; LF_i = ∅; C_i = I; C = I; l = 1
while |C| > 0 do
      // Find all valid candidates
      F_i* = {A ∈ C_i : supp_i(A) > k_i}
```

```
        // Broadcast candidates
        C = F_1^* ∪ · · · ∪ F_t^*
        // Global test
        F = F ∪ {B ∈ C : supp(B) > k}
        LF_i = LF_i ∪ {B ∈ C : B ∈ F, B ∈ F_i^*}
        // New local candidate set
        C_i = {B ∈ P(I) : |B| = l, A ⊆ B ⇒ A ∈ LF_i}
        l = l + 1
return F
```

Algorithm 2.4 Distributed Apriori algorithm

2.11 Summary

This section has explained on secure multi party computation and its weaknesses and strength. It can be seen that it is still popular to use a third party to perform secure multi-party computations due to increased efficiency over most encryption-based solutions, which are much more computationally costly. However there is now a large body of research in both Semi-trusted Third Party (STTP) solutions and Two-party solutions that involve data distortion. This demonstrates that there is a growing trend towards developing solutions that have a middle ground between efficiency and privacy. This chapter bring several SMC problems and their solutions to light such as database queries, intrusion detection, geometric computation, Statistical Analysis and Scientific Computation. Researches are still underway to get efficient solutions to all the SMC problems and as the scope of the SMC are growing wider and wider, this area is gaining a lot of interest and attention. With widespread use of computers, proliferation of sensitive and private data is very important. Next chapter describes the methodology adopted in this research.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

The process flow of the research is developed according to the established methodology of an online secure multi party computation. The first process is enhancing existing multi party computation algorithms for online purposes. As discussed and highlighted in Chapter 2, unsolved issues exist in the current offline secure multi party computation algorithms so make them inapplicable in case of online applications. A new algorithm is designed to obtain a secure, consecutive and online multi party computation that will be explained subsequently. In order to test the performance and applicability of the algorithm, a distributed frequent itemset mining is developed using the designed online multi party computation algorithm.

Let consider a real life motivating example where privacy is important issues. Credit card transactions are taking a large share of the payment systems worldwide and have led to a higher rate of stolen account numbers and subsequent losses by banks. Each bank has a database which stores both legitimate and fraudulent credit card transactions, so let consider distributed data mining for credit card fraud detection. The objective is to build a data mining model which will be used by automated fraud detector systems to prevent fraudulent transactions. This data mining model must be very general and accurate, because a mistake means great loss of money. In order to build accurate models of credit card fraud, a data mining system must have access to information about the fraudulent patterns of all banks (i.e., some types of fraud can have occurred only in one bank). The problem is that

banks are restricted by law (and also by competitive reasons) from sharing data about their customers with other banks. However, they may share “black-box” models; that is, they can share knowledge about fraudulent transactions, but not their data.

This chapter starts with multi party computation designed motivation in Section 3.2 followed by explanations of the research framework in Section 3.3, with detailing the solution of the online secret sharing process and consecutively approach. The Section continues with the framework of online distributed frequent itemset mining over multi business databases and ends with a brief summary in Section 3.4.

3.2 Research Motivation

The studies conducted over existing multi party computation algorithms show that the current multi party computation algorithms works offline or utmost semi-online, due to their quiddity of the offline secret sharing and sequentially problem. Regarding to a lot of online scopes, present algorithms are not applicable or at best is implementable in addition to some environmental condition or network consideration. As an example, a classic application known as e-voting and a new growth application named web polls, cannot work offline, because the essence of those applications is online. It is impossible to gather all participants together before the application starts and avoid new participants to join the process. It seems that an online secure consecutive multi party computation solves the problem.

Figure 3.1 shows the problem of existing offline multi party computation algorithms and their imposed restrictions on the issue.

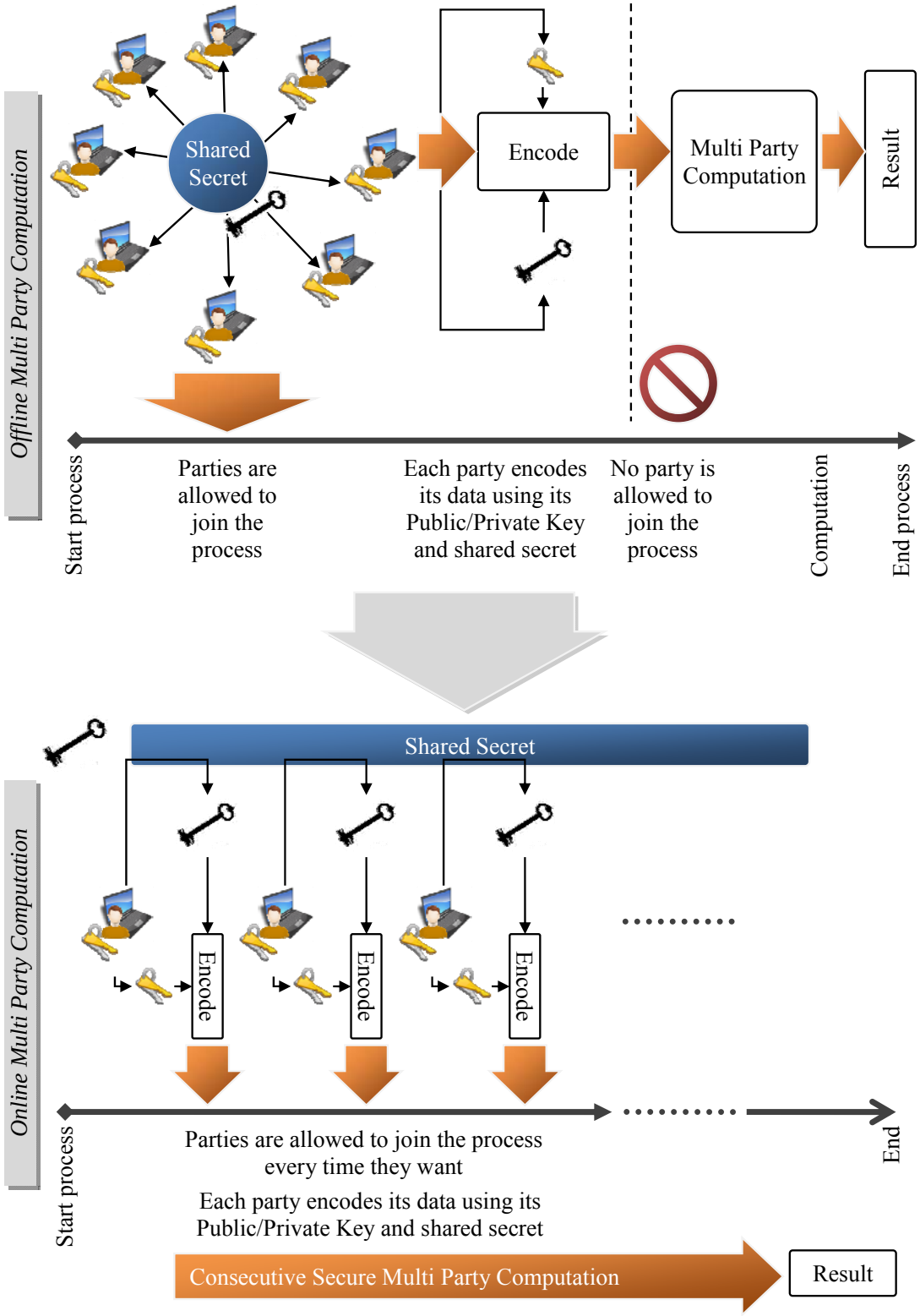


Figure 3.1 Research motivation

3.3 Research Framework

Figure 3.2 shows the online secure multi party computation framework employed in this research. It consists of four components; an Improved ElGamal based algorithm for online purposes, an online secret sharer using the infinite convergent product series, an consecutive multi party computation algorithm based on two party computation techniques and a online distributed privacy preserving frequent itemset mining tools over multi business databases.

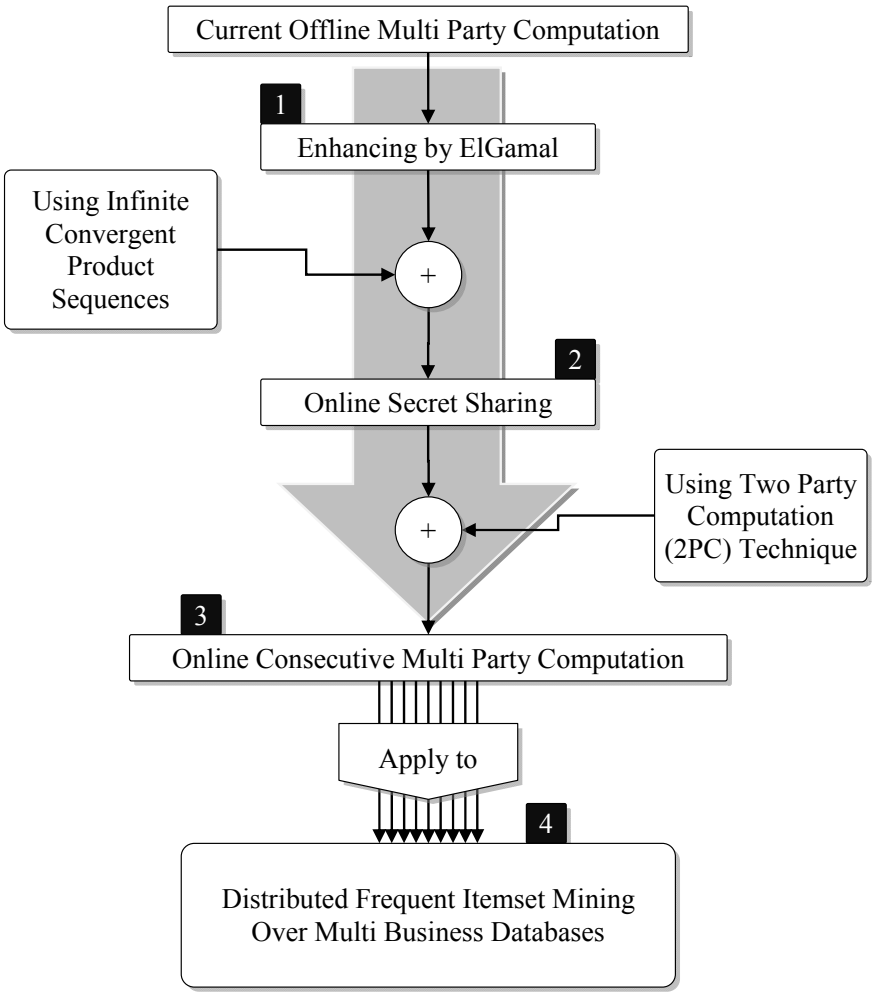


Figure 3.2 Research Framework

Based on the depicted framework, the research divides into four main phases. In first phases, current offline algorithms are investigated to adapt with online conditions. The second phase involved with designing a new online secret sharer against to classic offline secret sharing systems. The third phase concerned with

solving the main problem of consecutive multi party computation known as two party computation. While the online secure consecutive multi party computation algorithm is designed, a distributed frequent itemset mining develops using the innovative algorithm in the last phase. The research process is displayed in Figure 3.3 according to the four main phases.

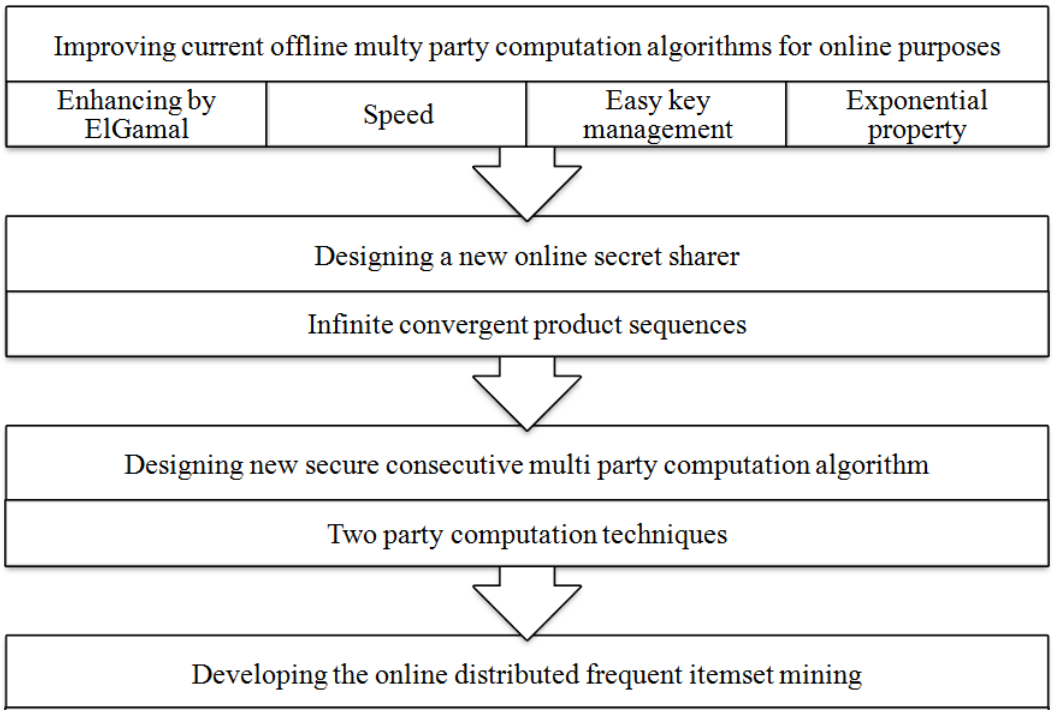


Figure 3.3 Research Procedure

The next sections focus on the details of the four main mentioned phases of the research procedure.

3.3.1 Online Parameters for Distributed Secure Multi Party Computation

In online cases, algorithm speed is one of the main factors, so the current algorithms should be enhanced to response faster or in fact should be finished in reasonable time. On the other hand, key management is the next problem in online applications. A method ensures easier key management, tends to better performance in online applications. The exponential property is the essential issue of the success distributed multi party computation. The fundamental trick of the secret sharing and

computing the function result over the distributed data is the exponential feature of the cryptography method which is used for encrypting the data. An ElGamal based multi party computation algorithm is developed to adapt with the online issues. The developed algorithm is fully described in Chapter 4. The main features which make it suitable for online purposes are listed below, briefly;

- Speed
- Easy key management and produce
- Guaranty the required exponential property

All steps are shown in Figure 3.4.

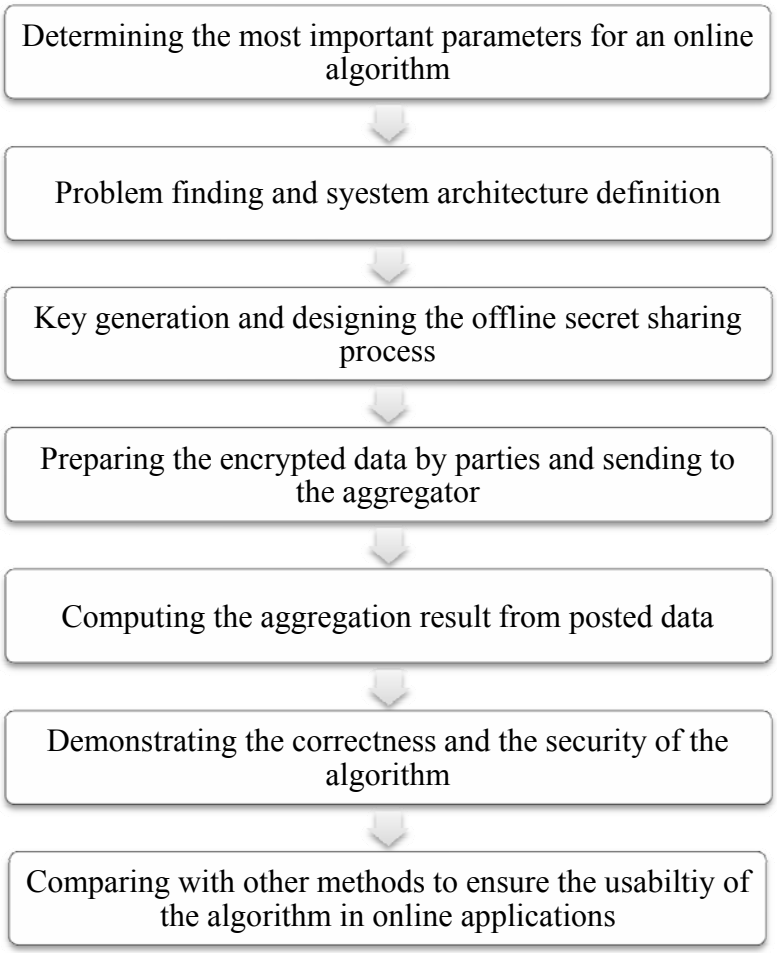


Figure 3.4 Enhancing offline multi party computation procedure

3.3.2 Online Secret Sharing

As mentioned before, current multi party computation algorithms use an offline secret sharing process to share a key among all parties. A new independence secret sharing approach is designed to be employed in online issues. The process of the new designed online secret sharing is depicted in Figure 3.5 and fully discussed in Chapter 4.4.

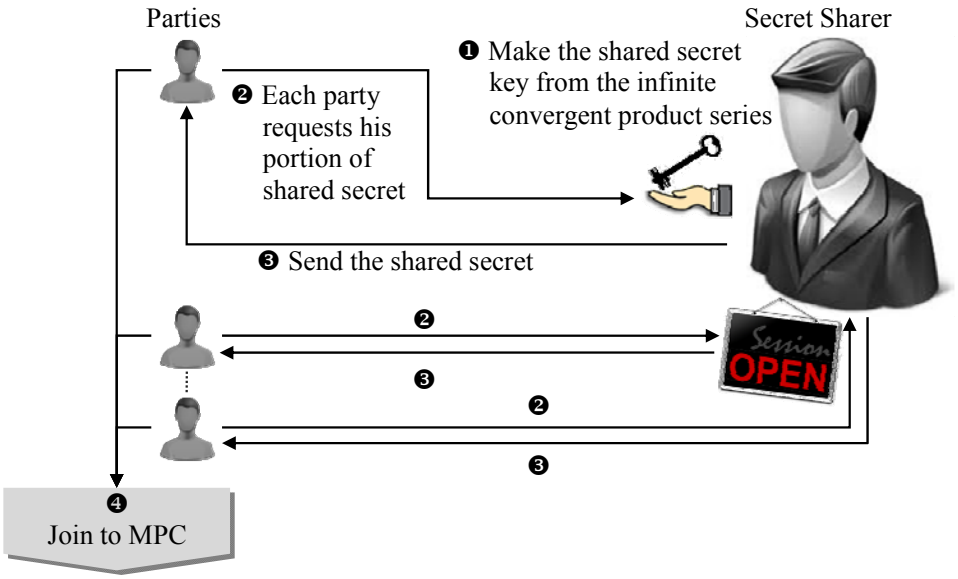


Figure 3.5 Online Secret Sharing

In the online secret sharing process, the dependent shared secret is replaced with the infinite convergent product series to make the session open during the multi party computation process. The online secret sharing process contains four main stages;

1. The secret sharer chooses an infinite convergent product series as the shared secret independently.
2. When a party joins to the process, he asks the secret sharer his piece of the shared secret key.
3. The secret sharer sends the portion of the key to participant.
4. The referred party joins the multi party computation process.

Because of independency between shared secret key and participants, the process can continue online. In addition, the infinity of the shared secret key ensures the unlimited participants' number.

The procedure of designing an online secret share is depicted in Figure 3.6.

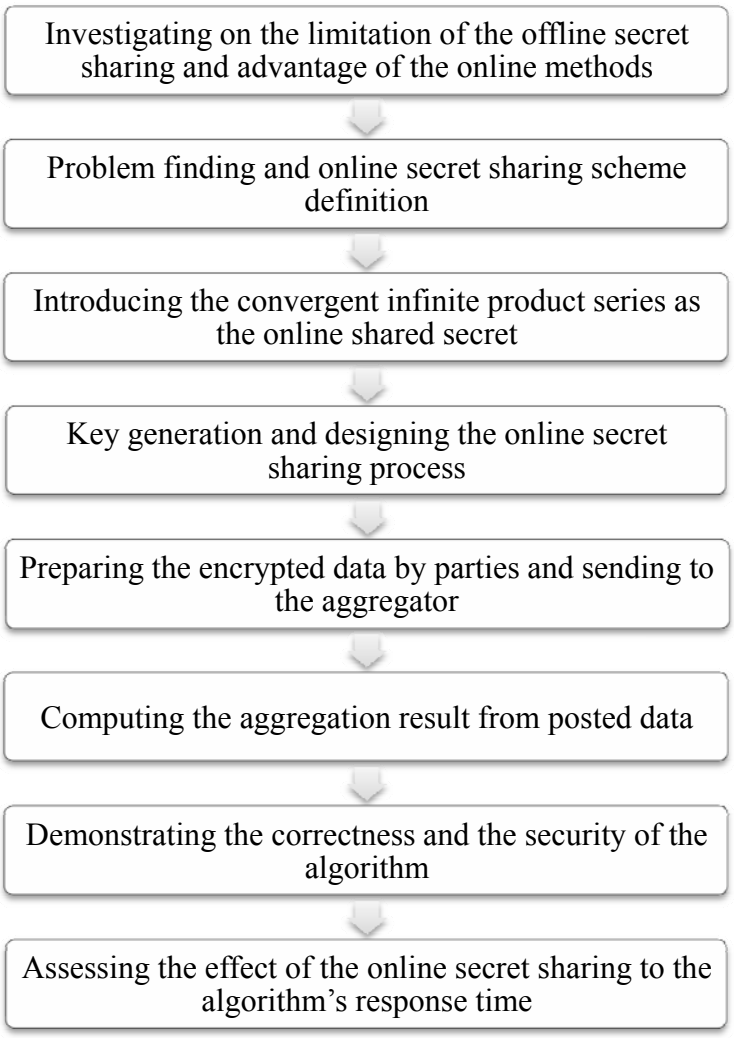


Figure 3.6 Designing online secret sharing procedure

3.3.3 Online Consecutive Computation

The next problem of transmission to online mode from offline mode is the computation consecutively. While the session is open forever and the number of participants who can join, are infinite; the computing engine should be enabled to

compute the result during the process.

Since the shared secret and participants' data are independence, despite of the parties' number, computing engine can calculate the result. The consecutive computation allows the adversary to find the parties' sensitive data. Assume that 100 parties send their data and the computing engine computes the aggregation result of those 100 parties. While the next party joins the system and sends his data, the computing engine computes the new result, consecutively. A simple calculation over these two results can reveal the new party's data. Without loosing the generalization, the problem is the classic issue of two party computation.

A new model is designed to solve the consecutively problem based on the two party computation techniques. The early steps of the process are same as before, but before encryption, each party uses the random number which produced by a two party randomizer. There are four components in the new model: the secret sharer, the 2PC randomizer, the computing engine and the aggregator. First each party asks his portion of the shared secret key from the secret sharer, and then computes his data with the shared secret key, random number produced by 2PC randomizer and his public keys.

The encrypted data is posted to the computing engine to compute the randomized encrypted result. The randomized encrypted result posts to the aggregator and finally aggregator decrypt the result by the randomize string produced by 2PC randomizer. Since to overcome the 2PC problem, the system needs at least three honest components, it is vital that the 2PC randomizer and the computing engine do not connect each other and be honest.

Whole step of the online consecutive multi party computation designing is shown in Figure 3.7.

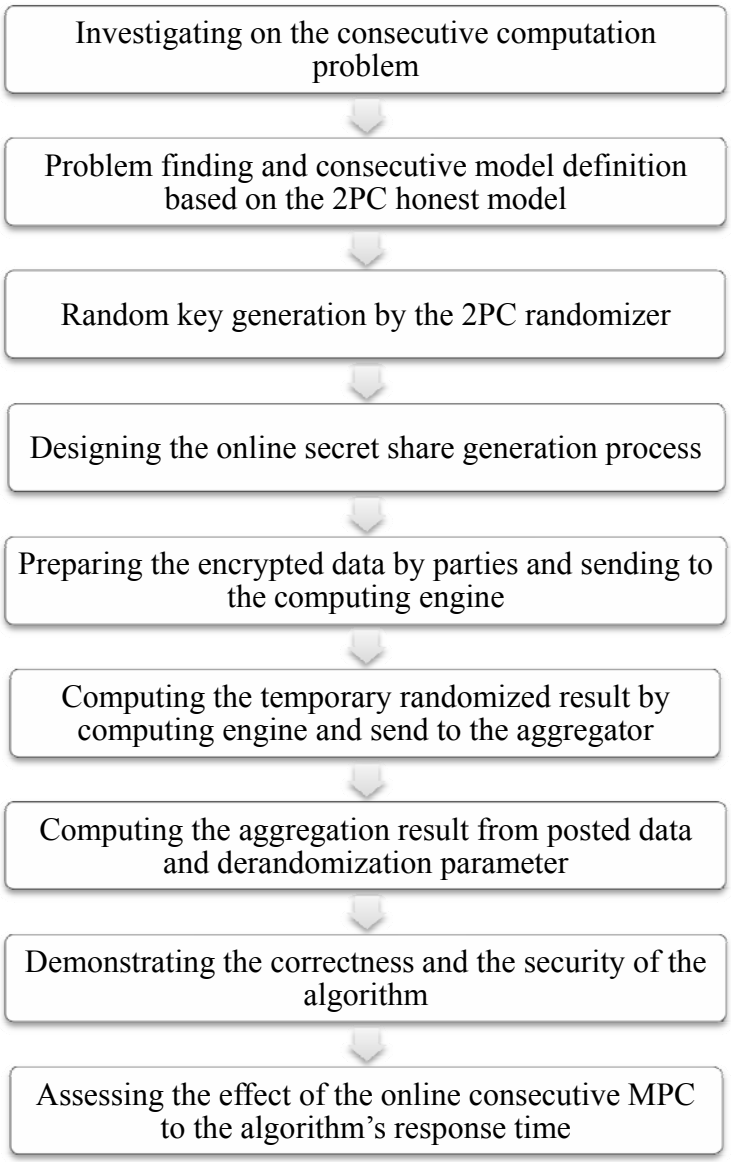


Figure 3.7 Online consecutive computation procedure

3.3.4 Distributed Frequent Itemset Mining

The distributed frequent itemset mining framework is shown in Figure 3.8. Suppose that there are $n \geq 3$ businesses and each company owns his private transaction database P_i. With the given support threshold, the aim of the algorithm is to find out all association rules among those transaction databases which satisfy the prearranged threshold. The algorithm should ensure that the businesses are not able to gain any information about other companies' transactions, their association rules and sensitive information, unless the association rules which distributed data miner

reveals to all businesses and such information may be learned from the result.

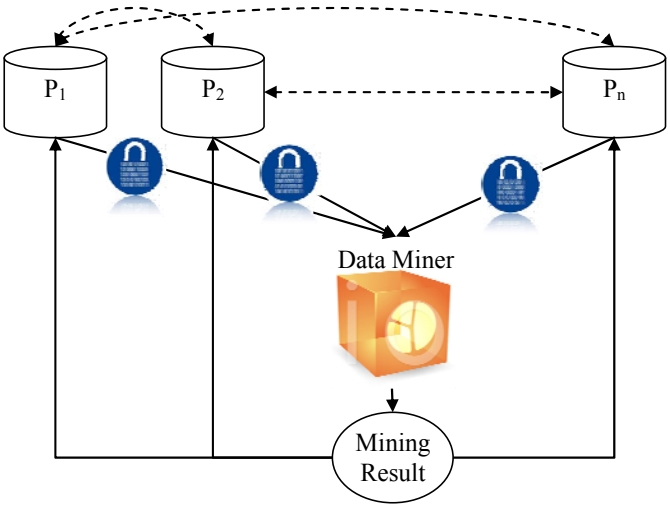


Figure 3.8 Distributed Frequent Itemset Mining

In the case of multi different businesses data mining, the data is partitioned vertically among the distributed databases and is related together with common fields. The distributed association rule mining over vertically partitioned databases tries to compute the support count of an itemset to decide whether the itemset is really frequent. The support count of the itemset can be securely computed using the innovative online multi party computation algorithm. If the securely computed support count is higher than the prearranged threshold, the miner marks the itemset as a frequent itemset.

Let us to consider a real life example. Figure 3.9 illustrates our example. There are some dataset in some hospitals that holds the patients' information. On the other hand, there are a lot of market transactions that show the people interesting to buy and use foods. These two different domains don't like to reveal their customers' data. However, they have a share field that relates their dataset together. This field is people identity. A good association rule mining over their data, leads to the roles that which people with which food addiction suffers from which disease. For example, the algorithm can find a rule $\{beef\_meat, sugar\} \Rightarrow \{diabetes\}$ that means most people use beef meat and sugar suffers from diabetes. In this case, there is a vertically partitioned data. Because each site's dataset is different with others, but they have a relational field that join their data together.

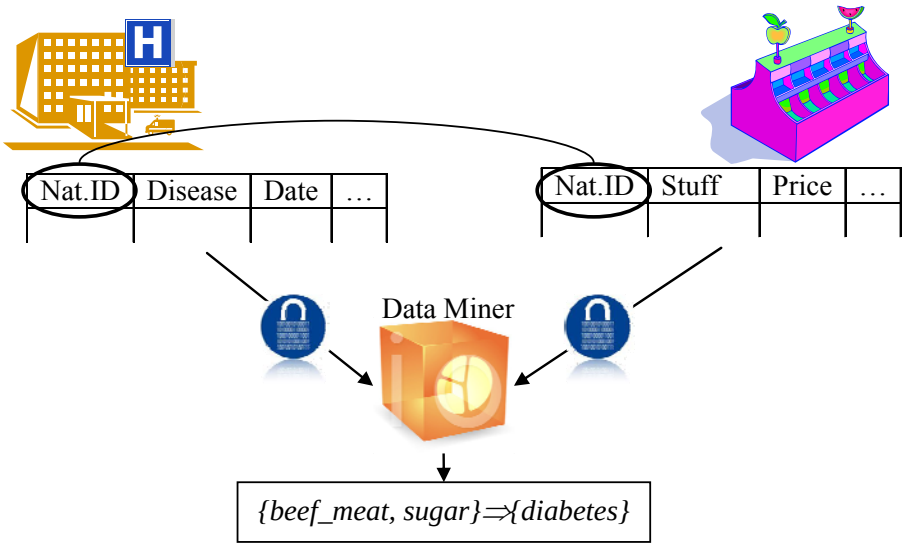


Figure 3.9 Frequent Itemset Mining over Two Vertically Partitioned Databases

The procedure of developing the secure distributed frequent itemset mining is depicted in Figure 3.10.

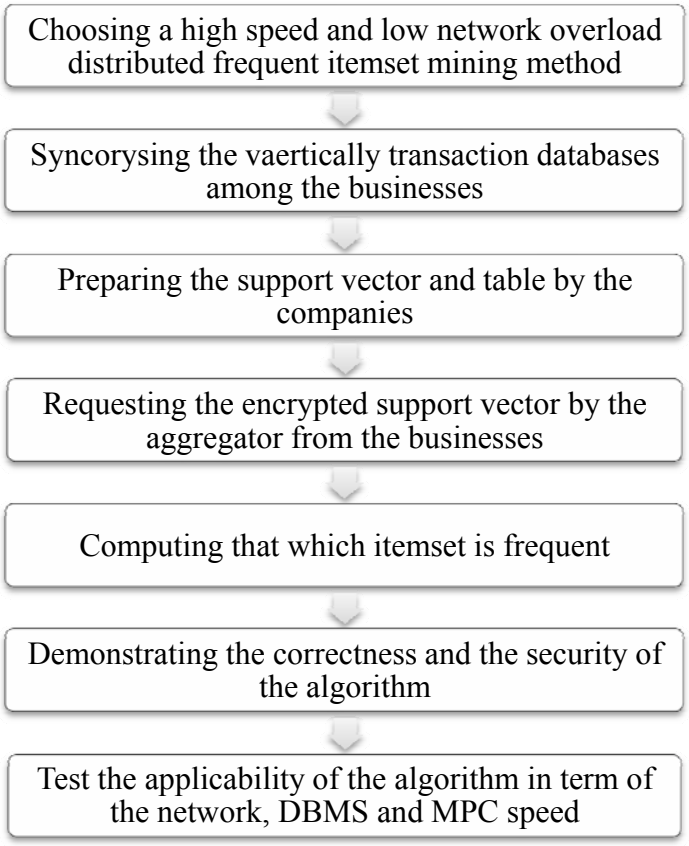


Figure 3.10 Secure distributed frequent itemset mining procedure

3.4 Summary

In this chapter, the research methodology has been described. In general, the approaches consist of enhancing the current offline multi party computation algorithms, the online secret sharing process, the consecutive computation and distributed frequent itemset mining. First, the research motivation is discussed, and the chapter is continued by the research framework. The lack of an online multi party computation is the fundamental issue of the research. The research is based on transition from offline mode to online mode. The first step is the improving the existing offline multi party computation to work efficient in the online framework. Two main problems of online multi party computation, the online secret sharing and the consecutive computation is discussed, and the new models are designed for solving the problems. The chapter is finalized with introducing of the distributed frequent itemset mining model as the application of the purposed online consecutive multi party computation algorithm. The next chapters involve to the fully description of the steps explained in the research methodology.

CHAPTER 4

A NEW ONLINE CONSECUTIVE SECURE MULTI PARTY COMPUTATION ALGORITHM

4.1 Introduction

In this chapter, a new cryptographic method is proposed to compute online secure consecutive aggregation of distributed data without collecting them in one place. The proposed algorithm does not reveal any sensitive content of data. The proposed algorithm is solved the problem of consecutive transaction in real case studies. In this algorithm, each client only sends a message to computing engine and there is no need to any inner interaction between clients. This feature decreases the chance of malicious attacks to access data and also decreases the processing time. The algorithm ensures that no extra information is revealed to all participants except the final computation of respondents' data.

The chapter continues by the description of enhanced ElGamal based multi party computation algorithm in section 4.2 and then online secret sharing method with infinite convergent product sequences is explained in section 4.3, overcoming to consecutively problem or two party computation is presented in section 4.4 and ends with a brief summary in section 4.5.

4.2 ElGamal Enhanced Multi Party Computation Algorithm

While an algorithm runs in online mode, there are important parameters which the algorithm designer should consider such as the speed, network bandwidth, computation space and user friendly.

For the first step, an improvement is employed to the current offline multi party computation scheme to runs in online mode efficiently. The ElGamal cryptosystem is selected as base as the developed algorithm. Most important factors that discriminates the ElGamal from others cryptosystem are listed below;

- Speed and Robustness
- Low network bandwidth usage
- Low computational complexity and overload
- Easy key producing and management

In addition the ElGamal cryptosystem guaranties the exponential property which is the fundamental element of multi party computation algorithms and is also secure under Diffie-Hillman DDH assumption (Chen *et al.*, 2008, Wang *et al.*, 2006).

The implemented algorithm is based on the homomorphism property of mentioned ElGamal encryption (Hirt and Sako, 2000). The DDH assumption and the ElGamal cryptosystem ensure the privacy of the algorithm. The algorithm also uses the exponentiation's mathematical properties for converting multiplication to desired sums. Modular arithmetic operation is used to speed up the computing time of big prime numbers. It is surely affect on algorithm time.

4.2.1 Algorithm Preliminaries

The system includes possibly large numbers of clients who own their private data and a computing engine who counts or sums their data. The architecture is shown in Figure 4.1. Although the computing engine should be able to count or sum the data, but the system need to protect privacy of clients' data. It means that the computing engine compute the summation of data without revealing the client's exact data to neither himself nor any other clients. So the clients used an encryption system to send their data to computing engine and the computing engine will compute the summation of their data without decrypting them.

In the proposed scenario, there are n clients, call them $C_1, \ldots, C_n$; respectively. Each client owns his private data d_i. The aim of the computing engine is to calculate the sum $d = \sum_{i=1}^{n} d_i$ with ensuring the privacy of d_i.

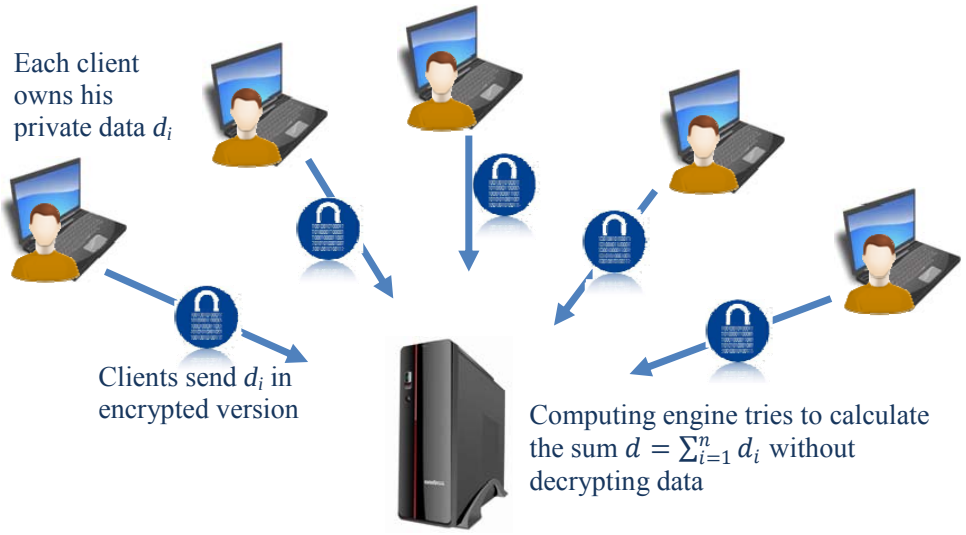


Figure 4.1 System Architecture

In the proposed model, because of its practicability, the clients do not need to know about other clients and they never communicate themselves. So there is no communication channel between different clients. Moreover, each client only sends one encrypted message to the computing engine. So, they do not need multi round interaction between clients and the computing engine.

4.2.2 Enhanced Offline Algorithm

Suppose that G is a group where $|G|=q$ and q is enough large prime), and assume that g generates the group G. The group G is assumed for all computations in this paper. In this algorithm each party C_i has two pairs of keys;

1. $((x_i \ mod_q), (X_i \ mod_q=(g^{x}{}_i \ mod_q))$
2. $((y_i \ mod_q), (Y_i \ mod_q=(g^{y}{}_i \ mod_q))$.

The shared secrets also define as

$$\left(X \ mod_q\right) = \prod_{i=1}^{n}\left(X_i mod_q\right) \quad (4.1)$$

$$\left(\mathrm{Y} \ \mathrm{mod_q}\right) = \prod_{\mathrm{i=1}}^{\mathrm{n}}\left(\mathrm{Y_i} \ \mathrm{mod_q}\right) \quad (4.2)$$

The values x_i and y_i are private keys; X_i and Y_i are public keys. All respondents need to calculate the values of X and Y from public keys.

First, each respondent C_i try to encrypt his value d_i using his private key x_i , y_i and shared public key X , Y in ElGamal encryption system as described below

$$E(d_i) = \begin{Bmatrix} \left(m_i \ mod_q\right) = \left(g^{d_i} \ mod_q\right).\left(X^{y_i} \ mod_q\right) \\ \left(h_i \ mod_q\right) = \left(Y^{x_i} \ mod_q\right) \end{Bmatrix} \quad (4.3)$$

Then all clients send their encrypted message to computing engine. The computing engine gathers all encrypted data together and computes m, h as:

$$m = \prod_{i=1}^{n}(m_i \ mod_q) \quad (4.4)$$

$$h = \prod_{i=1}^{n}(h_i \ mod_q) \quad (4.5)$$

The computing engine will use m, h for decrypting the d as sum of d_i. Then

the computing engine tries to find correct *d* between all possible values. It means that the computing engine tries to calculate ($g^d \ mod_p$) for all possible of *d* values. This stage is more time consuming step and will continue until there exist any *d* as

$$(m \ mod_q) = (g^d \ mod_q).(h \ mod_q) \qquad (4.6)$$

This value is the desired summation of respondents' data. The discrete logarithms cannot be employed by the computing engine; the computing engine must use trying and failing to find desired *d*. The time consumption parameter of algorithm returns to the range of possible values of *d*. In case of Boolean data the range of *d* is the number of respondents, *n*. The computing engine algorithm is shown in Figure 4.2.

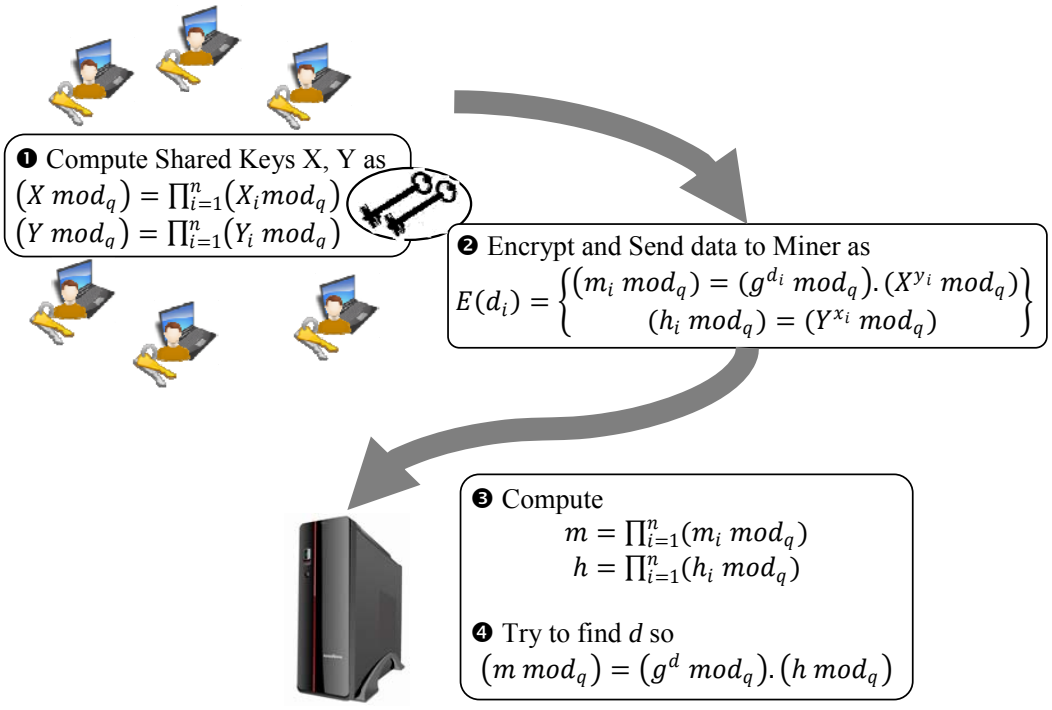


Figure 4.2 ElGamal based MPC procedure

The algorithm also is semi coded in Algorithm 4.1.

Input: Each party P_i has two pair of the public and private keys (x_i, $X_i = g^{x_i}$) and (y_i, $Y_i = g^{y_i}$) and his vital data d_i.

Outpout: Summation of all vital data $d=\sum_{i=1}^{n} d_i$.

Step 1.

Offline computing of the shared secret key X, Y

$$(X \bmod_q)=\prod_{i=1}^{n}(X_i \bmod_q),(Y \bmod_q)=\prod_{i=1}^{n}(Y_i \bmod_q)$$

Step 2. Sending data by the Participants

Each party encrypt his vital data

$$E(d_i)=\begin{Bmatrix}(m_i \bmod_q)=(g^{d_i} \bmod_q).(X^{y_i} \bmod_q)\\ (h_i \bmod_q)=(Y^{x_i} \bmod_q)\end{Bmatrix}$$

Send m_i , h_i to the aggregator.

Step 3. Computing the aggregation result by aggregator

Aggregator computes m, h

$$m=\prod_{i=1}^{n}(m_i \bmod_q),\ h=\prod_{i=1}^{n}(h_i \bmod_q)$$

For d = 1 to Max(d) do

If $(m \bmod_q)=(g^d \bmod_q).(h \bmod_q)$ then

return (d).

Algorithm 4.1 ElGamal Based offline multi party computation algorithm

4.2.3 Enhanced Offline Algorithm Demonstration

Suppose that $(m \bmod_q) = (g^d \bmod_q).(h \bmod_q)$

$$\Rightarrow \prod_{i=1}^{n}(m_i \bmod_q) = (g^d \bmod_q).\prod_{i=1}^{n}(h_i \bmod_q)$$

$$\Rightarrow (g^d \bmod_q) = \frac{\prod_{i=1}^{n}(m_i \bmod_q)}{\prod_{i=1}^{n}(h_i \bmod_q)}$$

$$= \frac{\prod_{i=1}^{n}(g^{d_i} \bmod_q).(X^{y_i} \bmod_q)}{\prod_{i=1}^{n}(Y^{x_i} \bmod_q)}$$

$$= \frac{\prod_{i=1}^{n}\left(g^{d_i}\ mod_q\right).\prod_{i=1}^{n}\left(X^{y_i}\ mod_q\right)}{\prod_{i=1}^{n}\left(Y^{x_i}\ mod_q\right)}$$

$$= \prod_{i=1}^{n}\left(g^{d_i}\ mod_q\right).\frac{\prod_{i=1}^{n}\left(X^{y_i}\ mod_q\right)}{\prod_{i=1}^{n}\left(Y^{x_i}\ mod_q\right)}$$

$$= g^{\sum_{i=1}^{n}(d_i\ mod_q)}.\prod_{i=1}^{n}\frac{\left(\left(\prod_{j=1}^{n}(X_j\ mod_q)\right)^{y_i}\ mod_q\right)}{\left(\left(\prod_{j=1}^{n}(Y_j\ mod_q)\right)\right)^{x_i}\ mod_q)}$$

$$= g^{\sum_{i=1}^{n}(d_i\ mod_q)}.\prod_{i=1}^{n}\frac{\left(\left(g^{\sum_{i=1}^{n}(x_j\ mod_q)}\right)^{y_i}\ mod_q\right)}{\left(\left(g^{\sum_{i=1}^{n}(y_j\ mod_q)}\right)\right)^{x_i}\ mod_q)}$$

$$= g^{\sum_{i=1}^{n}(d_i\ mod_q)}.\frac{g^{\sum_{i=1}^{n}\sum_{j=1}^{n}(x_j y_i\ mod_q)}}{g^{\sum_{i=1}^{n}\sum_{j=1}^{n}(y_j x_i\ mod_q)}} = g^{\sum_{i=1}^{n}(d_i\ mod_q)}$$

$$\Rightarrow \left(g^{d}\ mod_q\right) = g^{\sum_{i=1}^{n}(d_i\ mod_q)}$$

$$\therefore d = \sum_{i=1}^{n} d_i \qquad \blacksquare \text{ (4.7)}$$

In the last step, the aggregator tries to find d such that $\left(m\ mod_q\right) = \left(g^{d}\ mod_q\right).\left(h\ mod_q\right)$. The above demonstration proves that the found d is the aggregation of all participants' vital data d_i or $d = \sum_{i=1}^{n} d_i$.

4.3 Online Secret Sharing

Base on the offline ElGamal based algorithm represented in Section 4.3, the first and mostly important step is computing secret shared keys *X, Y*. the secret sharer needs all clients' public key to calculate those secret shared keys. Regarding to this situation, all clients should present their public keys before process starts. It means that if any of client's connection fails or any of clients does not join the process at start time, the secret sharer is not able to start process and algorithm will be

unsuccessful. The secret sharer should also know the exact number of clients. It leads to a semi-offline system as depicted in Figure 4.3.

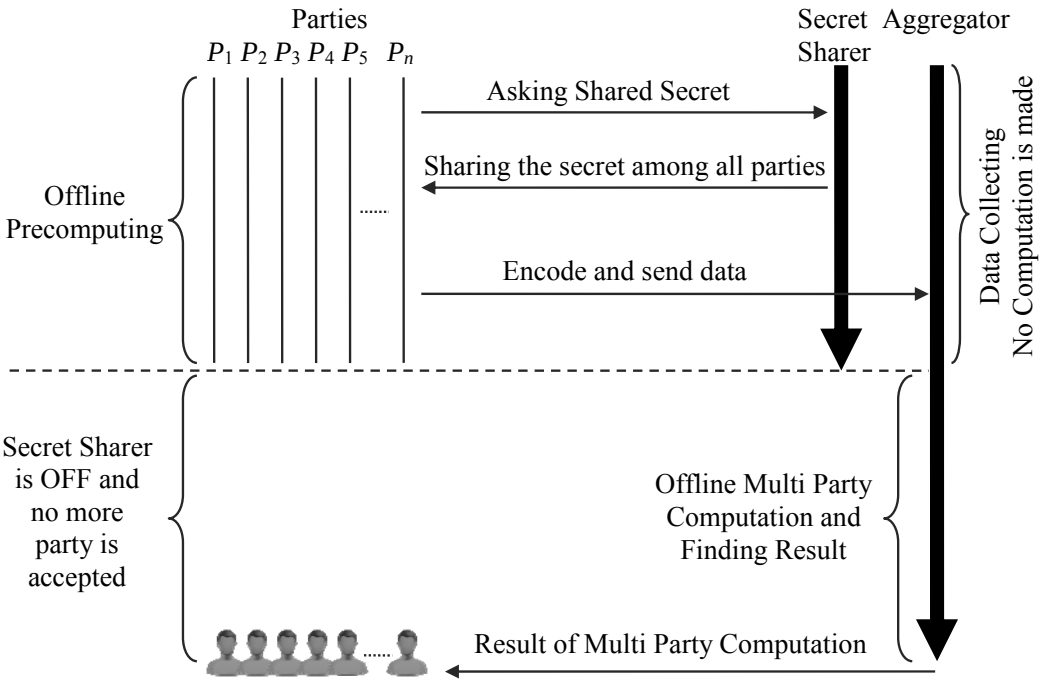


Figure 4.3 Offline secret sharing scheme

There are two separate phases in offline multi party computation algorithm. The first phase is offline secret sharing which the secret sharer tries to compute a shared secret dependent to the participants' public keys. Due to the dependency of the shared secret key to participants' public keys, only the parties can join who are present their portion of the data. While the secret sharer collects the participants' data and computes the shared secret key, no computation is allowed and the aggregator should wait until the offline precomputing finishes. After the shared secret computes and posts to all parties, the secret sharer completes his task and disappears. Moreover, no more party is accepted, because the new party does not contribute into creating the shared secret key. Although the encrypting and sending the encrypted data to aggregator can be processed any time, but due to the offline secret sharing process, the algorithm is not employable in online application. The algorithm behaves like semi-online in the best condition. This condition avoids using the algorithm in real world online problems like as E-Voting or web page's poll result. So, in most online cases the algorithm will fail and is inapplicable. Due to the computing engine, likewise, clients cannot assess the product of user's public keys

that do not join yet.

4.3.1 Algorithm Preliminaries

In these cases, the clients own their private and confidential data and the computing engine want to know the summation of all data without revealing the clients' secret data. But the clients are not ready at the beginning time and the time of user's appearance is optional. The computing engine also does not have any knowledge that how many clients could join to the system and when they will appear. So the secret sharer cannot compute the secret shared between the clients. As mentioned before, the secret shared between the users is the product of their public keys as called *X, Y*. All clients should be present their public keys to compute the shared secret key *X, Y* before aggregation started. Moreover, all users should send their data; therefore, the computing engine could compute the summation. The online secret sharing scheme is shown in Figure 4.4.

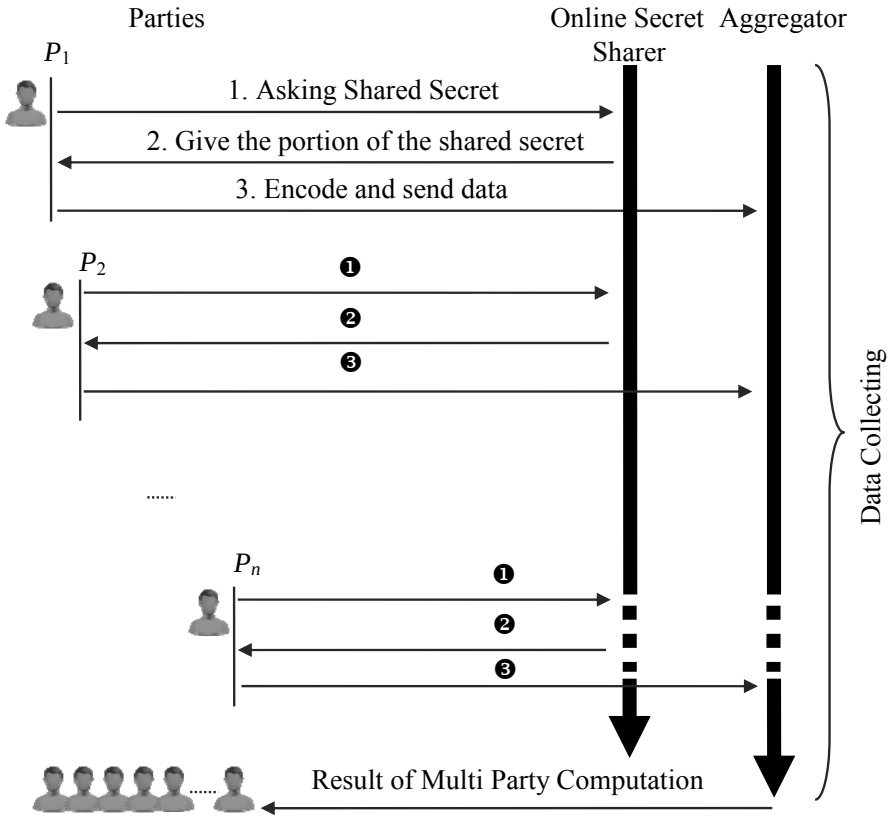


Figure 4.4 Online secret sharing scheme

In this model, the secret sharer is online during the process life and the users can join the system any time and ask the shared secret key. This model requires the independency between the shared secret key and users' public keys. While the participants receive the shared secret key, they try to encrypt their sensitive data with their public and private keys in addition to the shared secret key. The encrypted data is posted to the computing engine to be aggregated with others. At the end of the process, the result posts to all parties by the aggregator.

4.3.2 Online Secret Sharing algorithm

An infinite convergent product series is used as shared secret key instead of *X, Y* to establish the independency between the shared secret key and the participants' public keys. The final amounts of these series do not change with increasing the number of elements, so they ensure the unlimited number of clients who can join the process. An infinite product series define as

$$A = \prod_{i=1}^{\infty} \alpha_i \qquad (4.8)$$

like as

$$\prod_{i=1}^{\infty} \frac{1}{e} \left(\frac{1}{3i+1}\right)^{3i+\frac{1}{2}} = 1.0123785 \qquad (4.9)$$

Same as before, the proposed algorithm also is based on the homomorphism property of mentioned ElGamal encryption. The security of ElGamal encryption depends on new random values being used for each encryption. Each client C_i has two pairs of keys

1. $((x_i \ mod_q), (X_i \ mod_q = (g^{x}{}_i \ mod_q))$
2. $((y_i \ mod_q), (Y_i \ mod_q = (g^{y}{}_i \ mod_q))$.

as private and public keys, respectively. Remember that different x_i and y_i

values should be used in next runs of the algorithm. Since the secret sharer cannot compute the *X*, *Y*, So he uses two infinite series Ω, Ψ instead of *X*, *Y*.

$$\left(\Omega_i mod_q\right) = \left(g^{\omega_i} mod_q\right) \quad (4.10)$$

$$\left(\Omega\ mod_q\right) = \prod_{i=1}^{n}\left(\Omega_i mod_q\right) \quad (4.11)$$

$$\left(\Psi_i mod_q\right) = \left(g^{\psi_i} mod_q\right) \quad (4.12)$$

$$\left(\Psi\ mod_q\right) = \prod_{i=1}^{n}\left(\Psi_i mod_q\right) \quad (4.13)$$

The client C_i encrypt his value d_i in ElGamal encryption system using Ω, Ψ as described below

$$E(d_i) = \begin{Bmatrix} \left(m_i\ mod_q\right) = \left(g^{d_i}\ mod_q\right).\left(\Omega^{y_i}\ mod_q\right) \\ \left(h_i\ mod_q\right) = \left(\Psi^{x_i}\ mod_q\right) \end{Bmatrix} \quad (4.14)$$

and send his encrypted message in addition to his public keys X_i, Y_i to the aggregator. The aggregator computes *m*, *h*, *X*, *Y* same as before;

$$m = \prod_{i=1}^{n}\left(m_i\ mod_q\right) \quad (4.15)$$

$$h = \prod_{i=1}^{n}\left(h_i\ mod_q\right) \quad (4.16)$$

$$X = \prod_{i=1}^{n}\left(X_i\ mod_q\right) \quad (4.17)$$

$$Y = \prod_{i=1}^{n}\left(Y_i\ mod_q\right) \quad (4.18)$$

The aggregator tries to find desired *d* to satisfy below

$$\left(m\ mod_q\right).\left(X^{\sum_{i=1}^{n}\psi_i}\right) = \left(g^d\ mod_q\right).\left(h\ mod_q\right).\left(Y^{\sum_{i=1}^{n}\omega_i}\right) \quad (4.19)$$

The desired *d* is the summation of clients' data. The algorithm also is semi coded in Algorithm 4.2.

Input: Each party P_i has two pair of the public and private keys (x_i, $X_i = g^{x_i}$) and (y_i, $Y_i = g^{y_i}$) and his vital data d_i.

Outpout: Summation of all vital data $d=\sum_{i=1}^{n} d_i$.

Step 1. Made by Online Secret Sharer

Choose two convergent infinite product series as the shared secret

$$(\Omega_i \bmod_q)=(g^{\omega_i} \bmod_q)$$

$$(\Omega \bmod_q)=\prod_{i=1}^{n}(\Omega_i \bmod_q)$$

$$(\Psi_i \bmod_q)=(g^{\psi_i} \bmod_q)$$

$$(\Psi \bmod_q)=\prod_{i=1}^{n}(\Psi_i \bmod_q)$$

Online secret sharer sends Ω, Ψ to joint party

Step 2. Sending data by the Participants

Each party encrypt his vital data

$$E(d_i)=\left\{\begin{matrix}(m_i \bmod_q)=(g^{d_i} \bmod_q).(\Omega^{y_i} \bmod_q) \\ (h_i \bmod_q)=(\Psi^{x_i} \bmod_q)\end{matrix}\right\}$$

Send m_i , h_i to the aggregator.

Step 3. Computing the aggregation result by aggregator

Aggregator computes X, Y, m, h

$$(X \bmod_q)=\prod_{i=1}^{n}(X_i \bmod_q),(Y \bmod_q)=\prod_{i=1}^{n}(Y_i \bmod_q)$$

$$m=\prod_{i=1}^{n}(m_i \bmod_q),\ h=\prod_{i=1}^{n}(h_i \bmod_q)$$

For d = 1 to Max(d) do

If $(m \bmod_q).(X^{\sum_{i=1}^{n}\psi_i})=(g^{d} \bmod_q).(h \bmod_q).(Y^{\sum_{i=1}^{n}\omega_i})$ then

return (d).

Algorithm 4.2 Online secret sharing algorithm

4.3.3 Online Secret Sharing Algorithm Demonstration

Suppose that $\left(m\ mod_q\right).\left(X^{\sum_{i=1}^{n}\psi_i}\right)=\left(g^d\ mod_q\right).\left(h\ mod_q\right).\left(Y^{\sum_{i=1}^{n}\omega_i}\right)$

$$\Rightarrow\left(\prod_{i=1}^{n}(m_i\ mod_q)\right).\left(X^{\sum_{i=1}^{n}\psi_i}\right)$$

$$=\left(g^d\ mod_q\right).\left(\prod_{i=1}^{n}(h_i\ mod_q)\right).\left(Y^{\sum_{i=1}^{n}\omega_i}\right)$$

$$\Rightarrow\left(g^d\ mod_q\right)=\frac{\left(\prod_{i=1}^{n}(m_i\ mod_q)\right).\left(X^{\sum_{i=1}^{n}\psi_i}\right)}{\left(\prod_{i=1}^{n}(h_i\ mod_q)\right).\left(Y^{\sum_{i=1}^{n}\omega_i}\right)}$$

$$=\frac{\left(\prod_{i=1}^{n}\left(g^{d_i}\ mod_q\right).\left(\Omega^{y_i}\ mod_q\right)\right).\left(X^{\sum_{i=1}^{n}\psi_i}\right)}{\left(\prod_{i=1}^{n}(\Psi^{x_i}\ mod_q)\right).\left(Y^{\sum_{i=1}^{n}\omega_i}\right)}$$

$$=\frac{\left(\prod_{i=1}^{n}\left(g^{d_i}\ mod_q\right).\prod_{i=1}^{n}\left(\Omega^{y_i}\ mod_q\right)\right).\left(X^{\sum_{i=1}^{n}\psi_i}\right)}{\left(\prod_{i=1}^{n}\left(\Psi^{x_i}\ mod_q\right)\right).\left(Y^{\sum_{i=1}^{n}\omega_i}\right)}$$

$$=\prod_{i=1}^{n}\left(g^{d_i}\ mod_q\right).\frac{\prod_{i=1}^{n}\left(\Omega^{y_i}\ mod_q\right)}{\prod_{i=1}^{n}\left(\Psi^{x_i}\ mod_q\right)}.\frac{\left(X^{\sum_{i=1}^{n}\psi_i}\right)}{\left(Y^{\sum_{i=1}^{n}\omega_i}\right)}$$

$$=g^{\sum_{i=1}^{n}(d_i\ mod_q)}.\frac{\Omega^{\sum_{j=1}^{n}(y_j\ mod_q)}}{\Psi^{\sum_{j=1}^{n}(x_j\ mod_q)}}.\frac{\left(X^{\sum_{i=1}^{n}\psi_i}\right)}{\left(Y^{\sum_{i=1}^{n}\omega_i}\right)}$$

$$=g^{\sum_{i=1}^{n}(d_i\ mod_q)}.\frac{\left(\prod_{i=1}^{n}\left(\Omega_i mod_q\right)\right)^{\sum_{j=1}^{n}(y_j\ mod_q)}}{\left(\prod_{i=1}^{n}\left(\Psi_i mod_q\right)\right)^{\sum_{j=1}^{n}(x_j\ mod_q)}}.\frac{\left(X^{\sum_{i=1}^{n}\psi_i}\right)}{\left(Y^{\sum_{i=1}^{n}\omega_i}\right)}$$

$$=g^{\sum_{i=1}^{n}(d_i\ mod_q)}.\frac{\left(\prod_{i=1}^{n}\left(g^{\omega_i} mod_q\right)\right)^{\sum_{j=1}^{n}(y_j\ mod_q)}}{\left(\prod_{i=1}^{n}\left(g^{\psi_i} mod_q\right)\right)^{\sum_{j=1}^{n}(x_j\ mod_q)}}.\frac{\left(X^{\sum_{i=1}^{n}\psi_i}\right)}{\left(Y^{\sum_{i=1}^{n}\omega_i}\right)}$$

$$= g^{\sum_{i=1}^{n}(d_i \, mod_q)} . \frac{g^{\sum_{i=1}^{n}(\omega_i \, mod_q) . \sum_{j=1}^{n}(y_j \, mod_q)}}{g^{\sum_{i=1}^{n}(\psi_i \, mod_q) . \sum_{j=1}^{n}(x_j \, mod_q)}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= g^{\sum_{i=1}^{n}(d_i \, mod_q)} . \frac{g^{\sum_{j=1}^{n}(y_j \, mod_q)^{\sum_{i=1}^{n}(\omega_i \, mod_q)}}}{g^{\sum_{j=1}^{n}(x_j \, mod_q)^{\sum_{i=1}^{n}(\psi_i \, mod_q)}}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= g^{\sum_{i=1}^{n}(d_i \, mod_q)} . \frac{\left(\prod_{i=1}^{n}\left(g^{y_i} \, mod_q\right)\right)^{\sum_{i=1}^{n}(\omega_i \, mod_q)}}{\left(\prod_{i=1}^{n}\left(g^{x_i} \, mod_q\right)\right)^{\sum_{i=1}^{n}(\psi_i \, mod_q)}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= g^{\sum_{i=1}^{n}(d_i \, mod_q)} . \frac{\left(\prod_{i=1}^{n}\left(Y_i \, mod_q\right)\right)^{\sum_{i=1}^{n}(\omega_i \, mod_q)}}{\left(\prod_{i=1}^{n}\left(X_i \, mod_q\right)\right)^{\sum_{i=1}^{n}(\psi_i \, mod_q)}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= g^{\sum_{i=1}^{n}(d_i \, mod_q)} . \frac{Y^{\sum_{i=1}^{n}(\omega_i \, mod_q)}}{X^{\sum_{i=1}^{n}(\psi_i \, mod_q)}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$\Rightarrow \left(g^{d} \, mod_q\right) = g^{\sum_{i=1}^{n}(d_i \, mod_q)}$$

$$\therefore d = \sum_{i=1}^{n} d_i \qquad \blacksquare \quad (4.20)$$

The above demonstration proves that the represented algorithm correctly computes the summation of respondents' data and found *d* is the $\sum_{i=1}^{n} d_i$.

4.4 Consecutive Multi Party Computation

The algorithm is secure under the *semi-honest* model. In this model, although $n - 2$ respondents are malicious and assist the aggregator; the aggregator cannot reveal the exact data of two remained respondents. In case of $n - 1$ malicious respondents, the two party computation (2PC) problem is occurred. In fact, if the aggregator knows the data of the *n*–1 respondents, considering that the aggregator can calculate the sum of all data, it is easy to find the data of the last respondent with subtracting the total value from all other respondents' data.

In case of the consecutive algorithm, there are two types of the *semi-honest* model. First the algorithm is secure if at least two respondents be honest. Second one is that in implementing the consecutive transactions, at least two messages should send to the aggregator simultaneously. In fact, the send transaction should be performed when there are at least two encrypted data. The *semi-honest* model is destroyed if this fact is ignored. If messages send to the aggregator one by one, then the aggregator calculates each respondent value by subtracting the current total from the last total.

A simple way to solve the issue is buffering system should buffer the encrypted message of the respondent P_{i+1} by system or network tricks and wait for next respondent or more respondents to establish the *semi-honest* model. Without loosing the generality of the problem, if the number of respondents is odd, the system could send a zero data with the last respondent's data. In this section, the consecutively computation problem is formulated and solved using the two party computation techniques.

4.4.1 Algorithm Preliminaries

Consider that the P_i respondents are joined the system and send their encrypted data to the aggregator. The aggregator calculates the total value T_i using the online aggregation algorithm. If the respondent P_{i+1} joins the system and sends his encrypted data to the aggregator; the aggregator calculates his data d_{n+1} by subtracting $T_{i+1} - T_i$ without decrypting the respondent message. The issue is depicted in Figure 4.7. The problem is similar to the classic problem known as the two-millionaire problem. In the cryptographic computation domain, the issue named two party computation. As in highlighted in chapter 2, in two party computation, it is essentially impossible to calculate the summation of two different parties' data without revealing the exact value of each party. In fact, if a party knows the summation, due to the knowledge of his value, the party can compute another party's data. The computation does not need to the complex tasks or knowledge about the breaking of the cryptographic techniques, just need to learn how to subtract two

numbers.

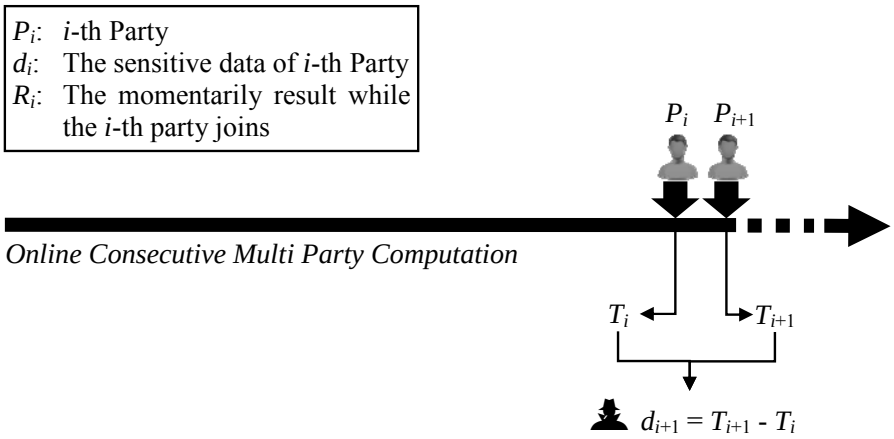


Figure 4.5 Consecutive computation reveals the confidential data

Without loosing the generality, if all parties from 1 to i is assumed as the one person namely *Alice*, and the party P_{i+1} is assumed as *Bob*, the problem is two party computation issue between two millionaires *Alice* and *Bob.*

The scenario is same as before which is explained in online secret sharing section. There are n participants P_1 to P_n , who are interested to join an online multi party computation process. They want to share their vital information d_1 to d_n together to compute a function $f(d_1, d_2, \ldots, d_n)$ by the computing engine. Although, all parties should learn the outcome, but it is important that no party can compute other parties' vital information.

The parties are allowed to join the process every time and the secret sharer opens the session until the end of the process. The addressed issue that should be solved is employing a technique that avoids the computing engine to compute the function consecutively which leads to the two party computation problem. The process should still ensure the security and confidentially of vital information posted by participants, unless the information reveals by the outcome of the function.

4.4.2 Online Consecutive Algorithm

Since, the two party problem is essentially unsolvable, to establish a secure model, at least three independent component is required. So there is no link or communication channel between the 2PC randomizer and the computing engine. The Third component is the aggregator. The model is shown in Figure 4.6.

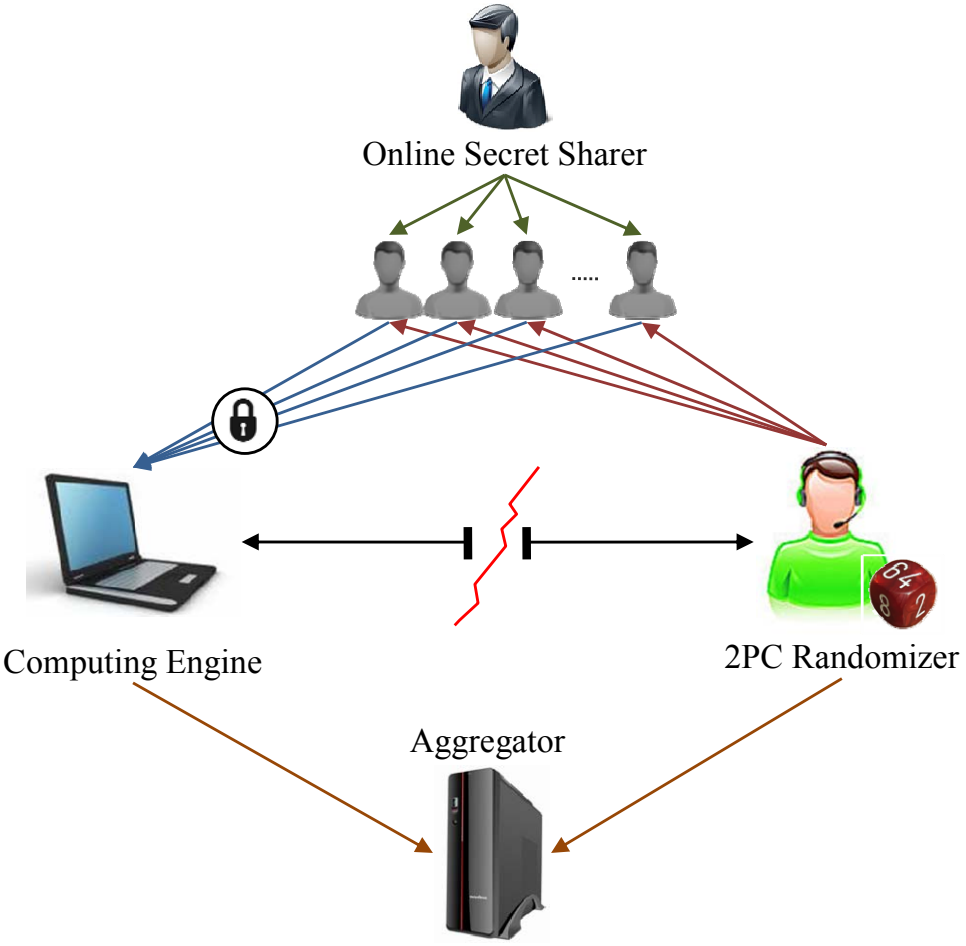


Figure 4.6 Online Consecutive Computation Model

The algorithm contains six main stages as follows;

1. Aggregator starts the process.
2. The 2PC randomizer initializes the set of random keys.
3. Each party who joins the process asks the 2PC randomizer a pair of

random keys and the shared secret key from the online secret sharer.

4. Encrypt the vital data using the random keys, the shared secret key and the owned public and private keys and sends the encrypted data to the computing engine.

5. The computing engine calculates the production of the posted data and sends it to the aggregator.

6. The aggregator decrypts the result using the 2PC randomizer parameters and ends the process.

4.4.3 Initializing Setup

The aggregator starts the process by sending the estimation number of participants, *n*, to the 2PC randomizer. The 2PC randomizer creates two sets of the random keys according to the number of participants.

$$1.\{(\vartheta_1, \theta_1 = g^{\vartheta_1}\ mod_q), (\vartheta_2, \theta_2 = g^{\vartheta_2}\ mod_q), .., (\vartheta_n, \theta_n = g^{\vartheta_n}\ mod_q)\}$$

$$2.\{(\lambda_1, \Lambda_1 = g^{\lambda_1}\ mod_q), (\lambda_2, \Lambda_2 = g^{\lambda_2}\ mod_q), .., (\lambda_n, \Lambda_n = g^{\lambda_n}\ mod_q)\}$$

The 2PC randomizer also computes the θ, Λ as follows;

$$(\theta\ mod_q) = \prod_{i=1}^{n}(\vartheta_i mod_q) \quad (4.21)$$

$$(\Lambda\ mod_q) = \prod_{i=1}^{n}(\lambda_i mod_q) \quad (4.22)$$

On the other hand, the online secret sharer selects two infinite convergent product sequences Ω, Ψ as the shared secret keys.

$$(\Omega_i mod_q) = (g^{\omega_i} mod_q) \quad (4.23)$$

$$\left(\Omega \; mod_q\right) = \prod_{i=1}^{n}\left(\Omega_i mod_q\right) \quad (4.24)$$

$$\left(\Psi_i mod_q\right) = \left(g^{\psi_i} mod_q\right) \quad (4.25)$$

$$\left(\Psi \; mod_q\right) = \prod_{i=1}^{n}\left(\Psi_i mod_q\right) \quad (4.26)$$

4.4.4 Joining and Encrypting Data

Same as before, each participant C_i has his own confidential data d_i and two pair of his public and private keys (x_i , X_i) and (y_i , Y_i). The participant C_i asks the secret sharer Ψ, Ω and requests the 2PC randomizer a pair of random keys (ϑ_i , θ_i) and (λ_i , Λ_i). The party randomizes the vital data d_i with two random keys as;

$$E(d_i) = T_i = \left(g^{d_i} \; mod_q\right).\left(\Lambda^{\vartheta_i} \; mod_q\right) \quad (4.27)$$

The participant tries to share his data with others using the shared secret keys in ElGamal cryptosystem as follows;

$$E(T_i) = \begin{Bmatrix} \left(m_i \; mod_q\right) = T_i.\left(\Omega^{y_i} \; mod_q\right) \\ \left(h_i \; mod_q\right) = \left(\Psi^{x_i} \; mod_q\right) \end{Bmatrix} \quad (4.28)$$

and posts the m_i , h_i , X_i and Y_i to the computing engine.

4.4.5 Computing the Randomized Production

The computing engine computes *m, h, X, Y* same as before;

$$m = \prod_{i=1}^{n}\left(m_i \; mod_q\right) \quad (4.29)$$

$$h = \prod_{i=1}^{n}\left(h_i \; mod_q\right) \quad (4.30)$$

$$X = \prod_{i=1}^{n}(X_i \, mod_q) \quad (4.31)$$

$$Y = \prod_{i=1}^{n}(Y_i \, mod_q) \quad (4.32)$$

The computing engine tries to find T as the production of all T_i. The computing engine use the try and error process to find a T that satisfies the Formula 4.36.

$$\left(m \, mod_q\right).\left(X^{\sum_{i=1}^{n}\psi_i}\right) = \left(T \, mod_q\right).\left(h \, mod_q\right).\left(Y^{\sum_{i=1}^{n}\omega_i}\right) \quad (4.33)$$

The desired T is the production of participants' randomized data. The computing engine sends the proposed T to the aggregator for derandomization process.

$$T = \prod_{i=1}^{n}(T_i \, mod_q) \quad (4.34)$$

Since, the computation is consecutive and because of the 2PC problem, the computing engine can compute the T_i, but he cannot derandomize the T_i to evaluate the confidential data d_i, since the computing engine does not knows the random keys associated with the participant by 2PC randomizer. Note that, the model ensures that no connection is existed between 2PC randomizer and the computing engine and at least one of them is honest.

4.4.6 Computing the Aggregation Outcome

The aggregator inquires the inverse random key R from 2PC randomizer and tries to derandomize the T using the key and finds the desired result. The inverse random parameter is;

$$R = \prod_{i=1}^{n}(\theta^{\lambda_i} \, mod_q) \quad (4.35)$$

The aggregator attempts to find a d such;

$$R = \left(g^{d}\ mod_{q}\right).T \qquad (4.36)$$

The found d is the required aggregation $d = \sum_{i=1}^{n} d_i$. In the next section, the correctness of the mentioned computations is proved.

The algorithm also is semi coded in Algorithm 4.3.

Input: Each party P_i has two pair of the public and private keys (x_i, $X_i = g^{x_i}$) and (y_i, $Y_i = g^{y_i}$) and his vital data d_i.

Outpout: Summation of all vital data $d=\sum_{i=1}^{n} d_i$.

Step 1. The Aggregator starts the process by sending estimated n to the 2PC randomizer and the online secret sharer.

Step 2. Made by Online Secret Sharer

Choose two convergent infinite product series as the shared secret

$$\left(\Omega_i \mathrm{mod}_q\right)=\left(g^{\omega_i} \mathrm{mod}_q\right)$$

$$\left(\Omega\ \mathrm{mod}_q\right)=\prod_{i=1}^{n}\left(\Omega_i \mathrm{mod}_q\right)$$

$$\left(\Psi_i \mathrm{mod}_q\right)=\left(g^{\psi_i} \mathrm{mod}_q\right)$$

$$\left(\Psi\ \mathrm{mod}_q\right)=\prod_{i=1}^{n}\left(\Psi_i \mathrm{mod}_q\right)$$

Online secret sharer sends Ω, Ψ to joint party

Step 3. Made by 2PC randomizer

The 2PC randomizer creates two sets of the random keys according to the number of participants.

$$\left\{\left(\vartheta_1, \theta_1=g^{\vartheta_1}\ \mathrm{mod}_q\right), \left(\vartheta_2, \theta_2=g^{\vartheta_2}\ \mathrm{mod}_q\right), .., \left(\vartheta_n, \theta_n=g^{\vartheta_n}\ \mathrm{mod}_q\right)\right\}$$

$$\left\{\left(\lambda_1, \Lambda_1=g^{\lambda_1}\ \mathrm{mod}_q\right), \left(\lambda_2, \Lambda_2=g^{\lambda_2}\ \mathrm{mod}_q\right), .., \left(\lambda_n, \Lambda_n=g^{\lambda_n}\ \mathrm{mod}_q\right)\right\}$$

Compute

$$\left(\theta\ \mathrm{mod}_q\right)=\prod_{i=1}^{n}\left(\vartheta_i \mathrm{mod}_q\right)$$

$$\left(\Lambda\ \mathrm{mod}_q\right)=\prod_{i=1}^{n}\left(\lambda_i \mathrm{mod}_q\right)$$

Sends Λ, ϑ_i to the i-th joint party.

Step 4. Sending data by the Participants

Each party encrypt his vital data

$$E(d_i)=T_i=\left(g^{d_i} \bmod_q\right).\left(\Lambda^{\vartheta_i} \bmod_q\right)$$

$$E(T_i)=\left\{\begin{matrix}\left(m_i \bmod_q\right)=T_i.\left(\Omega^{y_i} \bmod_q\right) \\ \left(h_i \bmod_q\right)=\left(\Psi^{x_i} \bmod_q\right)\end{matrix}\right\}$$

Send m_i , h_i to the aggregator.

Step 5. The aggregator ends the process

Aggregator sends the end message to the computing engine and the 2PC randomizer.

Step 6. Made by the Computing Engine

Computing engine computes X, Y, m, h

$$\left(X \bmod_q\right)=\prod_{i=1}^{n}\left(X_i \bmod_q\right),\left(Y \bmod_q\right)=\prod_{i=1}^{n}\left(Y_i \bmod_q\right)$$

$$m=\prod_{i=1}^{n}\left(m_i \bmod_q\right),\ h=\prod_{i=1}^{n}\left(h_i \bmod_q\right)$$

For T = 1 to Max(T) do

If $\left(m \bmod_q\right).\left(X^{\sum_{i=1}^{n}\psi_i}\right)=\left(T \bmod_q\right).\left(h \bmod_q\right).\left(Y^{\sum_{i=1}^{n}\omega_i}\right)$ then

return (T)

Computing engine computes sends the random temporary result T to the Aggregator

Step 7. Made by 2PC randomizer

2PC randomizer sends the derandomization parameters R to the aggregator

$$R=\prod_{i=1}^{n}\left(\theta^{\lambda_i} \bmod_q\right)$$

Step 8. Computing the aggregation result by aggregator

For d = 1 to Max(d) do

If $R=\left(g^{d} \bmod_q\right).T$ then

return (d).

Algorithm 4.3 Secure online consecutive multi party computation algorithm

4.4.7 Online Consecutive Algorithm Demonstration

The assertion is divided into two separate stages. First the computation is made by the computing engine is proven in 4.37 and then the calculation compiled by the aggregator is demonstrated in 4.38.

Prove 1.

Suppose that $\left(m\ mod_q\right).(X^{\sum_{i=1}^{n}\psi_i}) = \left(T\ mod_q\right).\left(h\ mod_q\right).(Y^{\sum_{i=1}^{n}\omega_i})$

$$\Rightarrow \left(\prod_{i=1}^{n}(m_i\ mod_q)\right).\left(X^{\sum_{i=1}^{n}\psi_i}\right)$$

$$= \left(T\ mod_q\right).\left(\prod_{i=1}^{n}(h_i\ mod_q)\right).(Y^{\sum_{i=1}^{n}\omega_i})$$

$$\Rightarrow \left(T\ mod_q\right) = \frac{\left(\prod_{i=1}^{n}(m_i\ mod_q)\right).(X^{\sum_{i=1}^{n}\psi_i})}{\left(\prod_{i=1}^{n}(h_i\ mod_q)\right).(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= \frac{\left(\prod_{i=1}^{n}\left(T_i\ mod_q\right).\left(\Omega^{y_i}\ mod_q\right)\right).(X^{\sum_{i=1}^{n}\psi_i})}{\left(\prod_{i=1}^{n}(\Psi^{x_i}\ mod_q)\right).(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= \frac{\left(\prod_{i=1}^{n}\left(T_i\ mod_q\right).\prod_{i=1}^{n}\left(\Omega^{y_i}\ mod_q\right)\right).(X^{\sum_{i=1}^{n}\psi_i})}{\left(\prod_{i=1}^{n}\left(\Psi^{x_i}\ mod_q\right)\right).(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= \prod_{i=1}^{n}\left(T_i\ mod_q\right).\frac{\prod_{i=1}^{n}\left(\Omega^{y_i}\ mod_q\right)}{\prod_{i=1}^{n}\left(\Psi^{x_i}\ mod_q\right)}.\frac{\left(X^{\sum_{i=1}^{n}\psi_i}\right)}{\left(Y^{\sum_{i=1}^{n}\omega_i}\right)}$$

$$= \prod_{i=1}^{n}\left(T_i\ mod_q\right).\frac{\Omega^{\sum_{j=1}^{n}(y_j\ mod_q)}}{\Psi^{\sum_{j=1}^{n}(x_j\ mod_q)}}.\frac{\left(X^{\sum_{i=1}^{n}\psi_i}\right)}{\left(Y^{\sum_{i=1}^{n}\omega_i}\right)}$$

$$= \prod_{i=1}^{n}(T_i \bmod_q) . \frac{\left(\prod_{i=1}^{n}(\Omega_i mod_q)\right)^{\sum_{j=1}^{n}(y_j \bmod_q)}}{\left(\prod_{i=1}^{n}(\Psi_i mod_q)\right)^{\sum_{j=1}^{n}(x_j \bmod_q)}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= \prod_{i=1}^{n}(T_i \bmod_q) \frac{\left(\prod_{i=1}^{n}(g^{\omega_i} mod_q)\right)^{\sum_{j=1}^{n}(y_j \bmod_q)}}{\left(\prod_{i=1}^{n}(g^{\psi_i} mod_q)\right)^{\sum_{j=1}^{n}(x_j \bmod_q)}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= \prod_{i=1}^{n}(T_i \bmod_q) . \frac{g^{\sum_{i=1}^{n}(\omega_i \bmod_q).\sum_{j=1}^{n}(y_j \bmod_q)}}{g^{\sum_{i=1}^{n}(\psi_i \bmod_q).\sum_{j=1}^{n}(x_j \bmod_q)}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= \prod_{i=1}^{n}(T_i \bmod_q) . \frac{g^{\sum_{j=1}^{n}(y_j \bmod_q)^{\sum_{i=1}^{n}(\omega_i \bmod_q)}}}{g^{\sum_{j=1}^{n}(x_j \bmod_q)^{\sum_{i=1}^{n}(\psi_i \bmod_q)}}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= \prod_{i=1}^{n}(T_i \bmod_q) \frac{\left(\prod_{i=1}^{n}(g^{y_i} \bmod_q)\right)^{\sum_{i=1}^{n}(\omega_i \bmod_q)}}{\left(\prod_{i=1}^{n}(g^{x_i} \bmod_q)\right)^{\sum_{i=1}^{n}(\psi_i \bmod_q)}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= \prod_{i=1}^{n}(T_i \bmod_q) . \frac{\left(\prod_{i=1}^{n}(Y_i \bmod_q)\right)^{\sum_{i=1}^{n}(\omega_i \bmod_q)}}{\left(\prod_{i=1}^{n}(X_i \bmod_q)\right)^{\sum_{i=1}^{n}(\psi_i \bmod_q)}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= \prod_{i=1}^{n}(T_i \bmod_q) . \frac{Y^{\sum_{i=1}^{n}(\omega_i \bmod_q)}}{X^{\sum_{i=1}^{n}(\psi_i \bmod_q)}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$\Rightarrow (T \bmod_q) = \prod_{i=1}^{n}(T_i \bmod_q)$$

$$\therefore T = \prod_{i=1}^{n} T_i \qquad \blacksquare \ (4.37)$$

The prove 1 demonstrates that the found T is random temporary result according to the product of the participants' randomized vital data or $\prod_{i=1}^{n} T_i = \prod_{i=1}^{n} E(d_i)$.

Prove 2.

Suppose that $R = \left(g^{d} \bmod_{q}\right).T$

$$\Rightarrow \prod_{i=1}^{n}(R_i \bmod_{q}) = \left(g^{d} \bmod_{q}\right).\prod_{i=1}^{n}(T_i \bmod_{q})$$

$$\Rightarrow \left(g^{d} \bmod_{q}\right) = \frac{\prod_{i=1}^{n}(T_i \bmod_{q})}{\prod_{i=1}^{n}(R_i \bmod_{q})}$$

$$= \frac{\prod_{i=1}^{n}\left(g^{d_i} \bmod_{q}\right).\left(\Lambda^{\vartheta_i} \bmod_{q}\right)}{\prod_{i=1}^{n}(\theta^{\lambda_i} \bmod_{q})}$$

$$= \frac{\prod_{i=1}^{n}\left(g^{d_i} \bmod_{q}\right).\prod_{i=1}^{n}\left(\Lambda^{\vartheta_i} \bmod_{q}\right)}{\prod_{i=1}^{n}\left(\theta^{\lambda_i} \bmod_{q}\right)}$$

$$= \prod_{i=1}^{n}\left(g^{d_i} \bmod_{q}\right).\frac{\prod_{i=1}^{n}\left(\Lambda^{\vartheta_i} \bmod_{q}\right)}{\prod_{i=1}^{n}\left(\theta^{\lambda_i} \bmod_{q}\right)}$$

$$= g^{\sum_{i=1}^{n}(d_i \bmod_{q})}.\prod_{i=1}^{n}\frac{\left(\left(\prod_{j=1}^{n}(\Lambda_j \bmod_{q})^{\vartheta_i} \bmod_{q}\right)\right)}{\left(\left(\prod_{j=1}^{n}(\theta_j \bmod_{q})\right)^{\lambda_i} \bmod_{q}\right)}$$

$$= g^{\sum_{i=1}^{n}(d_i \bmod_{q})}.\prod_{i=1}^{n}\frac{\left(\left(g^{\sum_{i=1}^{n}(\lambda_j \bmod_{q})}\right)^{\vartheta_i} \bmod_{q}\right)}{\left(\left(g^{\sum_{i=1}^{n}(\vartheta_j \bmod_{q})}\right)^{\lambda_i} \bmod_{q}\right)}$$

$$= g^{\sum_{i=1}^{n}(d_i \bmod_{q})}.\frac{g^{\sum_{i=1}^{n}\sum_{j=1}^{n}(\lambda_j\vartheta_i \bmod_{q})}}{g^{\sum_{i=1}^{n}\sum_{j=1}^{n}(\vartheta_j\lambda_i \bmod_{q})}} = g^{\sum_{i=1}^{n}(d_i \bmod_{q})}$$

$$\Rightarrow \left(g^{d} \bmod_{q}\right) = g^{\sum_{i=1}^{n}(d_i \bmod_{q})}$$

$$\therefore d = \sum_{i=1}^{n} d_i \qquad \blacksquare \text{ (4.38)}$$

Finally, the prove 2 demonstrates that the algorithm computes the aggregation

result of the parties' vital data correctly and securely.

4.4.8 Algorithm Security Analysis

The algorithm preserves the privacy of confidential data under the both *2PC-honest* and *user semi-honest* models. In the 2*PC-honest* model, there should exist more than two honest components. In the purposed algorithm, the aggregator, the 2PC randomizer and the computing engine are employed to establish the 2*PC-honest* model. If one of the proposed components is malicious, the algorithm ensures that the vital data cannot decrypt and is protected against the 2PC attacks. As mentioned before, in the *semi-honest* model, there are at least two honest parties and even though n -2 parties assist themselves; it is not possible to find the sensitive data of the two rest parties. The privacy fails, if the connection between the 2PC randomizer and the computing engine establishes or the components posts information together more than they should.

4.5 Distributed Frequent Itemset Mining

Classic data mining tries to collect all information in one central data warehouse and executing a data mining algorithm over the stored data warehouse. For example, the health department in contribution with all clinical and hospitals want to discover the patterns and diseases trends among the patients, such as forecasting an epidemic virus. On the other hand, the social institutes also own the good information about people's behavior who are living in target area. In addition, the businesses, trade companies or even hypermarkets can prepare considerable information about the habits of mentioned patients. The main issue is collecting all these information in a central data warehouse due to the data privacy. A classic way to face the privacy problem is to provide the resorted statistic data in each side, which does not refer to the patients' individual information, and then each side sends these resorted data to the trusted third party who builds the data warehouse and runs the data mining algorithm. It is obvious that today's business rules do not allow the

parties to join this process, although, the result patterns are much attractive. The purposed online consecutive multi party computation can be employed to address the mentioned issue. The new model is shown in Figure 4.7.

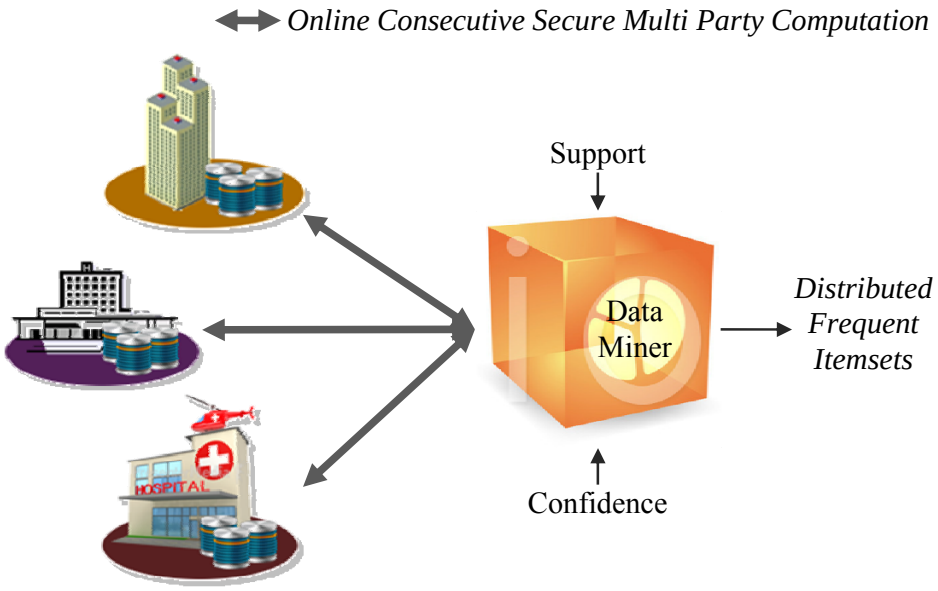


Figure 4.7 Frequent itemset mining with online consecutive secure multi party computation; Model and framework

In this model, the data is distributed vertically between some business and companies, and the data miner want to mine theses distributed data. Each party sends his data in encrypted version which nobody can decrypt it. The purposed secure multi party computation is employed by the data miner to compute the required calculation and finding the frequent itemset among the distributed information.

As highlighted and discussed in Chapter 2, the Apriori frequent itemset mining algorithm is selected as the base as the privacy preserving distributed frequent itemset mining.

4.5.1 Algorithm Preliminaries

There are N businesses who own their private transaction databases TR^i. Each

company precomputes his Boolean itemset matrix M^i from his transaction database. The Boolean itemset matrix is used for counting the support of itemsets. The width of the matrix *m* equals to the number of transaction database fields F^i (columns) and the height of the matrix *n* equals to the number of transaction database rows. Without loosing the generality, assume that all transaction databases include equivalent number of the rows. $M^i(k, j)$ denotes the existence of the $item_j$ in $transaction_k$ among the *i*th party's database. Table 4.1 shows a simple example of computing the Boolean itemset matrix.

Table 4.1 Example of the Boolean itemset matrix

Transaction #	$item_1$	$item_2$	…	$item_m$
1	*		…	*
2		*	…	*
3		*	…	
…	…	…	…	…
N	*	*	…	

→ M^i

$$\begin{bmatrix} 10 & 1 \\ 01 & 1 \\ 01 & 0 \\ \multicolumn{2}{c}{\cdots} \\ 11 & 0 \end{bmatrix}$$

Remember that the support count *SC* of an itemset $IS \subseteq \{item_j\}_{1 \le j \le m}$ is the number of transactions that includes all items belong to the itemset *IS*. Suppose that *FS* is the set of fields' number which belong to the itemset *IS*.

$$FS = \{j | item_j \in IS\} \quad (4.39)$$

The $transaction_k$ counts the itemset *IS*, if all $M_i(k, j)$ for all $j \in FS$ be 1, or multiplication of all those values equals to 1.

$$SC^i = \sum_{k=1}^{n} \prod_{j \in FS} M^i(k, j) \quad (4.40)$$

Since, each business owns his private transaction database and the fields of databases are different, some common fields should exist between all databases. This model is also known as the vertically distributed database. The businesses should sort their transaction databases base on the common fields privately. Te target transaction can determine by the miner or can be agreed by the companies. Back to examples,

the national ID of common people can be select for the common field between the hospitals, the health institutes, the social department and the hypermarkets who want to join the mining process.

After sorting the data base on the common fields, the miner tries to find the support count of the itemset *IS*. Since the data is distributed vertically among the all parties, each company owns the portion of the itemset *IS*. Assume that FS^i denotes the intersection between the selected itemset *IS* and the fields of the *i*th business transaction database F^i.

$$FS^i = IS \cap F^i = \{j | item_j^i \in F \ \wedge \ item_j \in IS\} \quad (4.41)$$

$$\cap_{i=1}^{N} FS^i = \emptyset \quad (4.42)$$

Let $x^i = \{x_1^i, x_2^i, \dots, x_n^i\}$ a Boolean support vector counted over the *i*th party Boolean itemset matrix regarding to the intersected itemset FS^i.

$$x_k^i = \prod_{j \in FS^i} M^i(k,j) \quad (4.43)$$

If the selected itemset supports by all business common transaction number *k*, then the summation of the x_k^i's for $1 \le i \le n$ equals to the number of parties *N*;

$$y_k = \begin{cases} 1 & \sum_{i=1}^{N} x_k^i = N \\ 0 & \sum_{i=1}^{N} x_k^i \neq N \end{cases} \qquad 1 \le k \le n \quad (4.44)$$

$$y_k = \begin{cases} 1 & \sum_{i=1}^{N} \prod_{j \in FS^i} M^i(k,j) = N \\ 0 & \sum_{i=1}^{N} \prod_{j \in FS^i} M^i(k,j) \neq N \end{cases} \qquad 1 \le k \le n \quad (4.45)$$

The miner wishes to count (y_k = 1) and compare the result with assessed support threshold. Recall that if the support count is greater than the support threshold then the itemset is frequent and inversely if the support count is lower than the support threshold then the itemset is not frequent.

$$y = \frac{\sum_{k=1}^{n} y_k}{n} \times 100 = \frac{\sum_{k=1}^{n}\left(\sum_{i=1}^{N} \prod_{j \in FS^i} M^i(k,j)=N\right)}{n} \times 100 \quad (4.46)$$

$$IS = \begin{cases} is\ frequent & y > confidence \\ is\ not\ frequent & y < confidence \end{cases} \quad (4.47)$$

4.5.2 Privacy Preserving Distributed Frequent Itemset

Based on the distributed Apriori algorithm, the miner starts with itemsets which include only one member, then combine frequent one member itemsets to find the two members itemsets and so on. The major idea is summarized below;

- To find frequent itemsets with *M* items, the miner first checks all itemsets with *M*-1 members that is frequent or not.
- The miner calculate all *l*-itemsets for *l* = 1 to *M*.

***Precomputation*:** The miner indicates the target community and sends it to all businesses. According to the target community, each company sorts its transaction databases for synchronization. The support threshold is assessed by the miner or all parties. Every business computes the Boolean itemset matrix M^i ($1 \le i \le N$) from the sorted transaction database. All precomputations perform privately in local databases and do not disclose to the other companies. The Boolean itemset matrix also computes locally and is private and no one access it unless the owner. The algorithm does not need to transfer any sensitive information such as patients' health information or clients' shopping list. The miner only requires that an itemset is supported by a transaction or not. The computation is performed by the business locally and the Boolean result are posted to the miner in encrypted version using the designed secure multi part computation algorithm, even the miner cannot compute each party's result. The miner only can calculate the summation of all parties' result.

***Frequent Itemset Mining*:**

1. The miner generates all *l*-itemsets IS_l for all *l* from 1 to the number of union fields *M*.

2. For an itemsets IS_l, the miner computes the FS^i denotes the intersection between parties' fields F^i and IS_l.

3. The miner sends FS^i to the according company.

4. In the next step, each company computes the Boolean support vector $x^i = \{x_1^i, x_2^i, \dots, x_n^i\}$ for desired FS^i.

5. The miner securely computes the summation of the x_k^i ($1 \le i \le N$) for all *k* from 1 to *n*, separately using the designed secure multi party computation algorithm.

6. The miner adjusts y_k as 1 if the mentioned summation equals to *N* and inversely as 0 if the mentioned summation is less than *N*.

7. The miner computes the summation of all y_k for *k* from 1 to *n*.

8. If the summation is greater than the support threshold value, then IS_l is frequent, otherwise, the IS_l is not frequent.

9. The non frequent itemsets are ignored. The miner increases the value of *l* and generates new itemsets IS_l with combination of the frequent itemsets.

10. For all new itemsets IS_l go to the step 2.

The algorithm continues until there is only one frequent itemset. Since there is no possible combination, the algorithm ends. The algorithm is summarized in Algorithm 4.4.

Input Parameter:

```
N: number of parties

M: number of all fields

n: number of target community
```

F^i:i^{th} party's fields set

m^i:$|F^i|$

TR^i: i^{th} party's transaction database

sp: Support threshold value

Precomputation

Miner sends the target community to all parties;

Each party sorts its transaction database based on

target community;

Each party computes the Boolean itemset matrix privately

$$M^i(k,j)=\begin{cases}1 & \text{if } TR^i(k) \text{ includes } item_j \\ 0 & \text{if } TR^i(k) \text{ does not include } item_j\end{cases} \quad \begin{matrix}1\le i\le N \\ 1\le k\le n; \\ 1\le j\le m^i\end{matrix}$$

l = N;

Step 1 - By Miner

IS_1 = Generate all possible l-itemset;

For each IS_1 do

For i = 1 to N do

$$FS^i=IS_1\cap F^i=\{j\,|\,item_j^i\in F \wedge item_j\in IS_1\}$$

Miner sends the FS^i to i^{th} party;

Step 2 - By Parties

Each party computes the Boolean support vector

$$x^i=\{x_1^i,x_2^i,\ldots,x_n^i\}\ ,\ x_k^i=\prod_{j\in FS^i} M^i(k,j)\ ;$$

Each party encrypts the x_k^i using the designed multi

party computation MPC;

Each party sends all E-MPC(x_k^i) to the Miner;

Step 3 - By Miner

Miner uses the MPC aggregation algorithm to compute

$$S_k=\sum_{i=1}^{N} x_k^i \qquad 1\le k\le n;$$

$$y_k=\begin{cases}1 & S_k=N \\ 0 & S_k\neq N\end{cases} \qquad 1\le k\le n;$$

The miner computes the summation of the y_k percentage

$$y = \frac{\text{D-MPC}\left(\sum_{k=1}^{n} y_k\right)}{n} \times 100;$$

If y > sp then

    IS_1 is frequent;

Else

    IS_1 is non frequent;

    Delete IS_1;

If |{IS_1| IS_1 is frequent}| = 1 then

    Go to END;

Else

    l = l+ 1;

    Go to Step 1;

END.

Algorithm 4.4 Privacy Preserving Distributed Frequent Itemset Algorithm

The algorithm computes *n* multi party computation aggregation for each itemset IS_l.

The algorithm is secure based on the designed multi party computation algorithm. Recall that, all condition for the secure multi party computation should establish for ensuring the security of the algorithm. The algorithm is protected under the 2*PC* and *semi honest* model under the DDH assumption.

4.6 Summary

In this chapter, the current offline multi party computation has been selected as the base of the proposed algorithm. The first attempt is improving the current offline algorithm to represent the better performance in the online modes. An ElGamal based multi party computation has been developed, due to its speed, less overloads, low complexity and easy key management. Two main problems are faced

to the existing offline multi party computation algorithms; the offline secret sharing and the consecutive computation. The first problem has been fully explained and solved by designing a new online secret sharer. The designed online secret sharing process uses the infinite convergent product series to make the process presents the shard secret keys all the process-life. The second problem has been solved by establishing the 2*PC-honest* model. The 2PC randomizer has been employed to design the online consecutive secure multi party computation algorithm. The chapter ends by describing the security and confidentiality of the purposed algorithm. The mathematic demonstration has been proven the privacy of the participants' vital data. Thus the proposed method is suitable for online application.

CHAPTER 5

E-VOTING AND DISTRIBUTED FREQUENT ITEMSET MINING, EXPRIMENTAL RESULTS AND CASE STUDIES

5.1 Introduction

The designed online consecutive secure multi party computation has the useful potential ability to solve the real online problems. Some experiments are performed to evaluate the performance of the algorithm. Two main case studies are implemented as the experimental applications; first e-voting and web polls, second the distributed frequent itemset mining. Both investigational applications employ the innovative online consecutive secure multi party application as base as their encrypted data transfer protocol.

5.1.1 Test Environment Architecture

The algorithm is implemented in Delphi. All cryptographic operations use the OpenSSL and FGBigInt libraries. The OpenSSL library is accessible at www.openssl.org website. The FGBigInt library is employed for generating the cryptographic keys and also to compute all cryptographic tools such as ElGamal computation, summation, multiplication, exponential and modular computation over the big integers. The library also is used to create the big prime numbers that is too important for any cryptographic systems.

The IIS HTTP Server is used for network simulation under windows Vista on a PC with a 2.4GHz processor and 2GB memory. The 512 bits are chosen as the length of each cryptographic key.

In this chapter, first the e-voting and web pools process are described in section 5.2. Three versions of the algorithm; ElGamal based offline multi party computation is explained in Section 5.2.1, continues with online secret sharing in Section 5.2.2 and finally online consecutive multi party computation are tested and discussed in Section 5.2.3. The distributed frequent itemset mining is explained and the result is introduced in Section 5.3.

5.2 E-Voting and Web Polls

You as a citizen may join the elections during your life, like as president or senate election. The classic process of election and counting all votes is a so time and cost consuming process. Today, all developed countries leans to modern E-Voting systems. On the other hand, almost all sites use polls to know their users' opinion. You can see polls in all around World Wide Web. The most important criteria in counting votes is voters' privacy preserving. It means that, even tough, the teller system should be able to count the summation of votes, but he shouldn't able to know the exact private opinion of a voter. Because of the votes are distributed in many parties and ensuring votes' confidentially, the algorithm sends the encrypted votes to Teller System and computes the poll result over multi parties.

In this research, a cryptographic method is proposed to allow a teller system to compute frequencies of respondents' votes. As mentioned before, it is important that content of votes should not be revealed. In this algorithm, each respondent only sends a message to teller system and there is no need to any inner interaction between respondents. The algorithm ensures that no extra information is revealed to Teller System except the frequency of respondents' votes. The offline e-voting algorithm is implemented as base as the developed ElGamal based multi party computation highlighted in chapter 4. The algorithm is fully depicted in Figure 5.1.

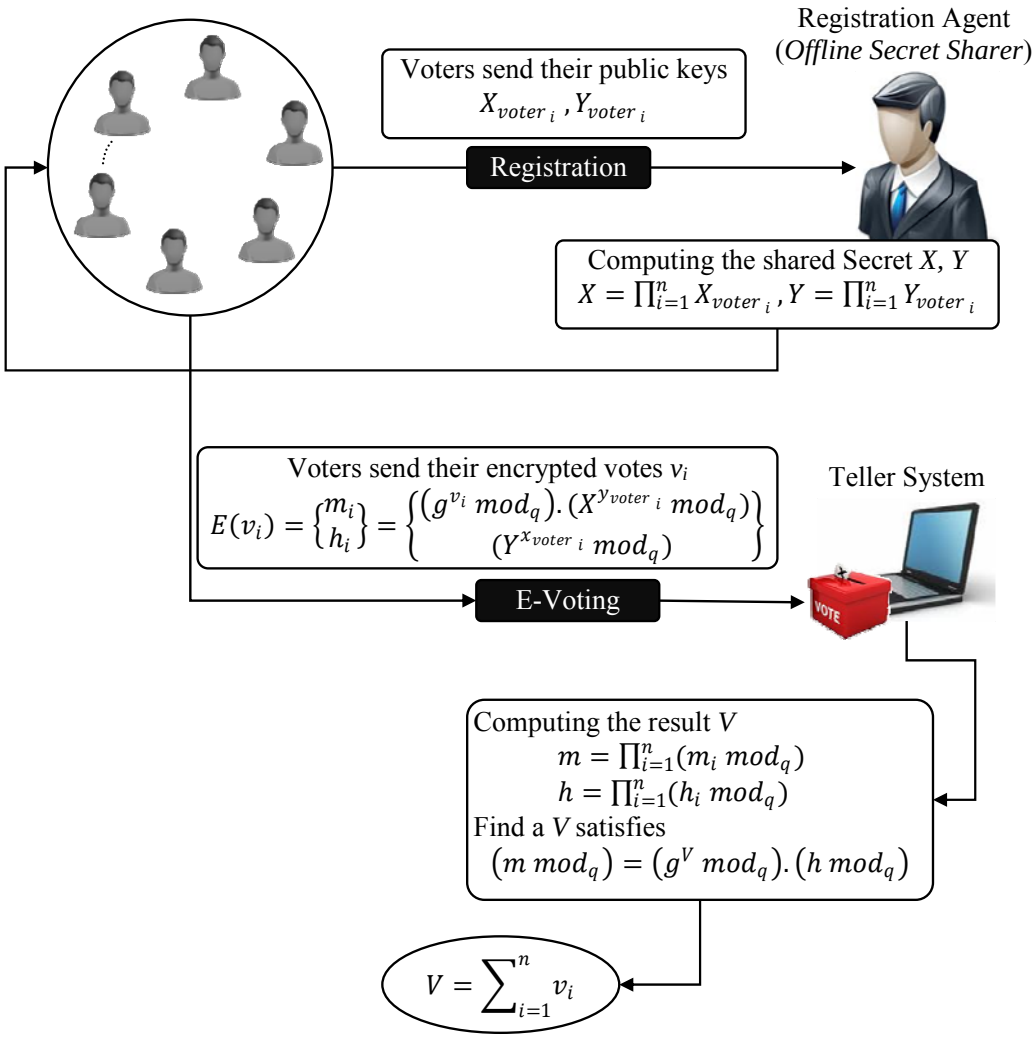


Figure 5.1 E-Voting using offline secret sharer

5.2.1 Offline E-Voting Experimental Result

The main parameters of experiment are: Number of respondents (Voters) and the range of *V*. Moreover, the time of computing vote v_i in each respondent will affect on final time. The key generation time and computing the algorithm parameters *X* and *Y* for each respondent also is time consuming process, but these two last parameters are ignored in the experimental result. Because, as mentioned before, these values can be precomputed offline before the protocol starts.

In our result, the time consumption of algorithm proposes as the main factor. The main stages are:

- Computing $E(v_i)$ on respondent U_i
- Sending Encrypted message to Teller
- Computing *m, h* on Teller
- Finding desired *V* among its' possible values

The first step is independent from the parameters; number of the voters and the range of *V*, because this step is local and self sufficient in each respondent. Therefore, this step ignores in the final time result. Next two factors are depending to the number of voters. The final result with enforce on these factors is listed in Table 5.1 as displayed in Figure 5.2.

Table 5.1 Processing time for different number of voters

Number of Voters	**50**	**100**	**200**	**500**	**1000**
Send Message	27	29	33	36	39
Computing m, h on Teller	2	12	26	91	93
Finding the result V	58	107	145	305	558
Total	87	148	204	432	690

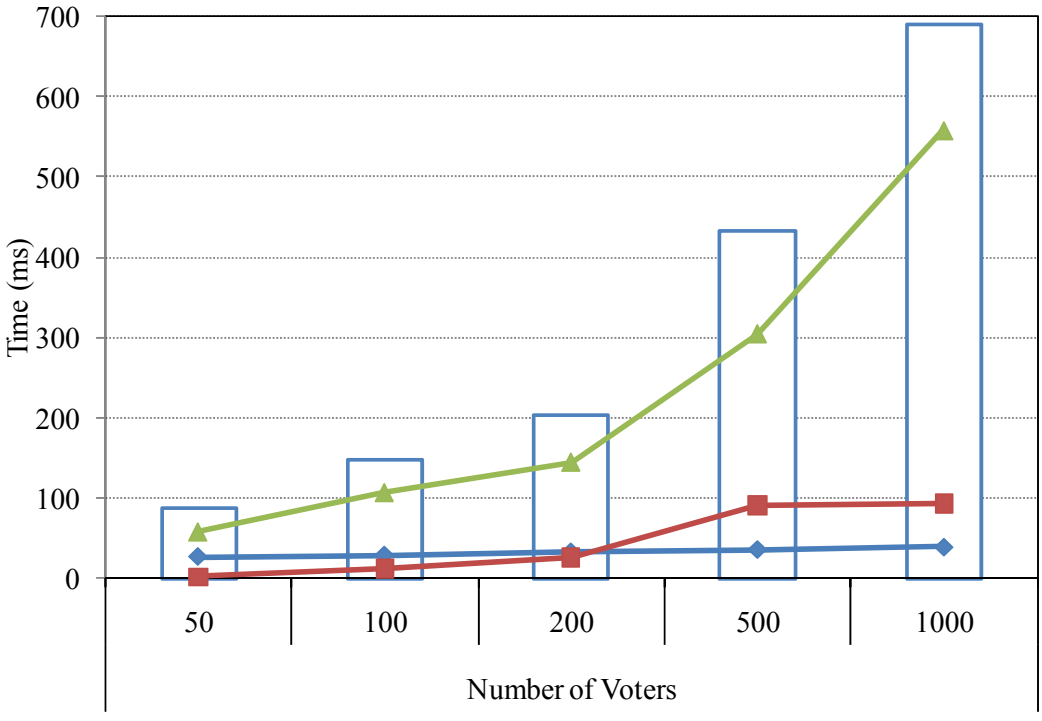


Figure 5.2 Total time of E-Voting process related to Number of respondents

In this experiment, the respondents send the Boolean vote to Teller. This model uses to exterminate the last parameter, because in this condition, the range of *d* is equal to number of voters. 50, 100, 200, 500 and 1000 voters are simulated in the experiment and the earned time is base on the average of five algorithm runs. The time offers a linear behavior related to the number of voters. The total time of the algorithm is shown by bars. For example, Teller computation takes 690 milliseconds for 1000 voters.

On the other hand, observing the finding desired summation loop on the Teller shows that in non Boolean cases, the time is highly related to the range of *V*. The range of *V* is not a simple parameter and return to many conditions. It also is impressible from the domain of vote's options. The Figure 5.3 shows the time of algorithm execution related to the number of voters and the range of *V*. The algorithm lasts less than 70 second for 1000 voters with the *V* in range of 1 to 1,000,000.

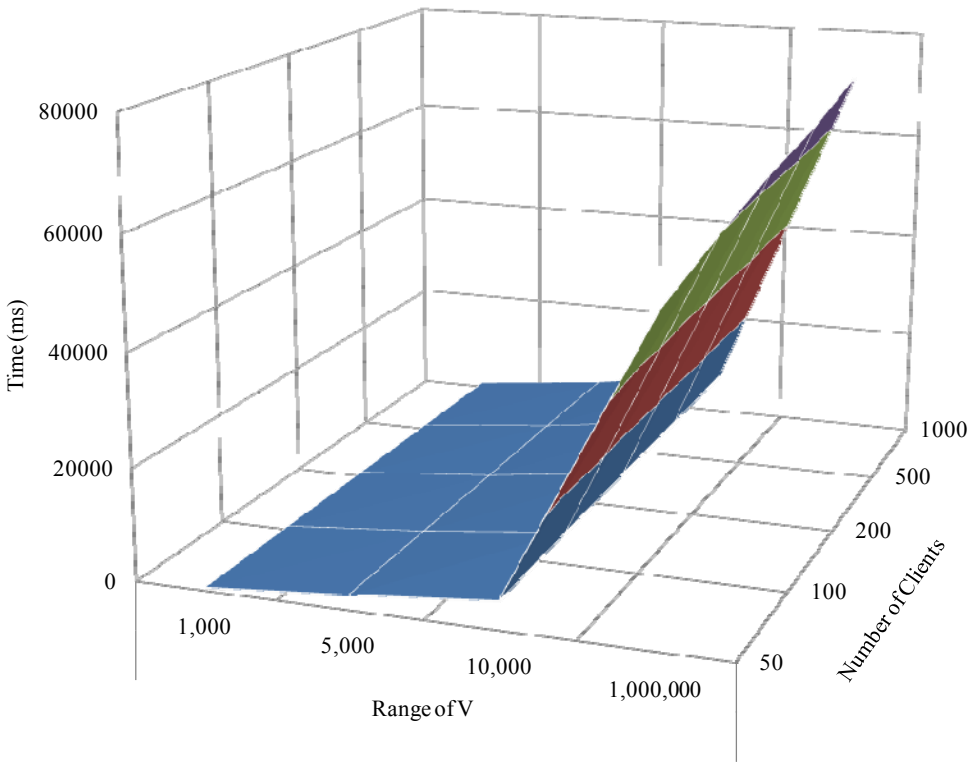


Figure 5.3 Total time related to Number of Voters and Range of *V*

Finally, the algorithm is compared with other homomorphic methods to find the suitable technique for online proposes. Two more homomorphic crypto system Paillier and RSA is selected for this experiment. As mentioned before, the most important parameter for online application which is focused in this experiment, include: speed, the easy key management and network overhead.

The overall time of the developed algorithm using ElGamal, Paillier and RSA homomorphic crypto system for variant number of voters in the range of 50 to 1000 voters are shown in Figure 5.4. Due to the low complexity computation of the ElGamal cryptosystem, this technique shows the best response time among all proposed techniques. Worst performance belongs to RSA, because of its highest complexity computation.

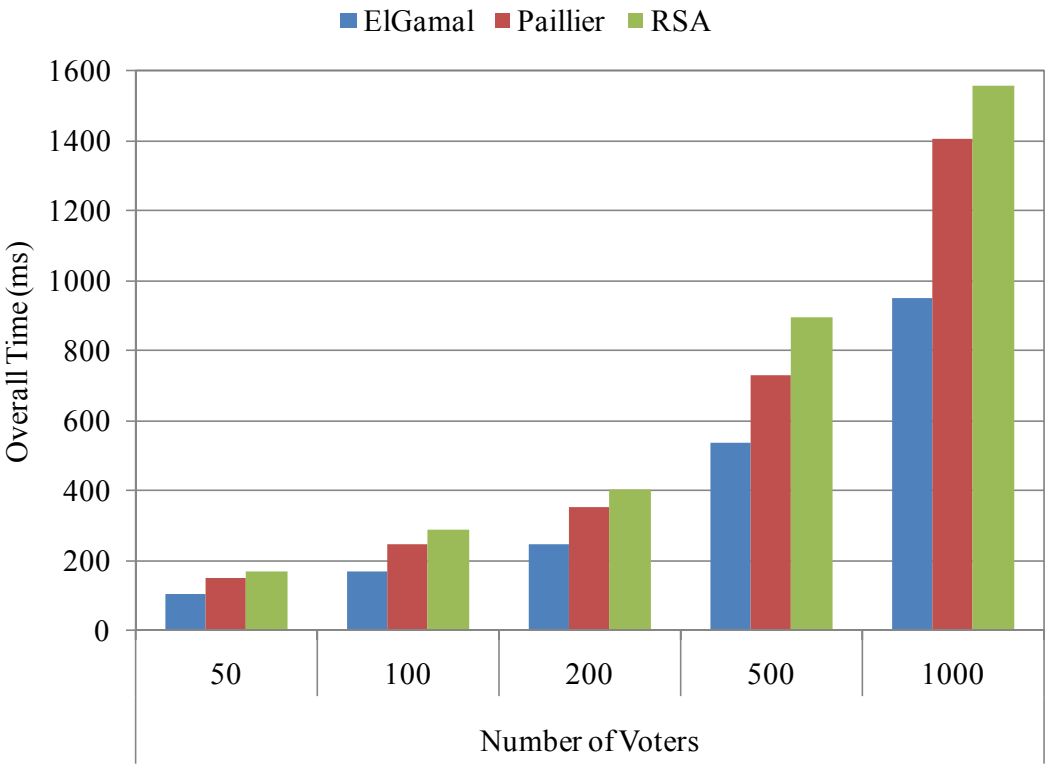


Figure 5.4 Overall response time of the proposed homomorphic methods

In term of the easy key management and network overhead, average of the key generation time and the posted package size are investigated as shown in Figure 5.5. These two parameter do not varies while the number of voters changes, because each voters sends and manage his keys individually an independently himself.

Although the times are not noticeably different among the all methods, but again the ElGamal shows the better performance related to the Paillier and RSA. Because the public key in ElGamal system is computed directly from the private key dependent, whereas in RSA two keys are generated parallel in a longer time process. The Paillier keys also are generated from the bigger range ($\mathbb{Z}_{n^2}$ versus by $\mathbb{Z}_n$) and it causes the longer generation time.

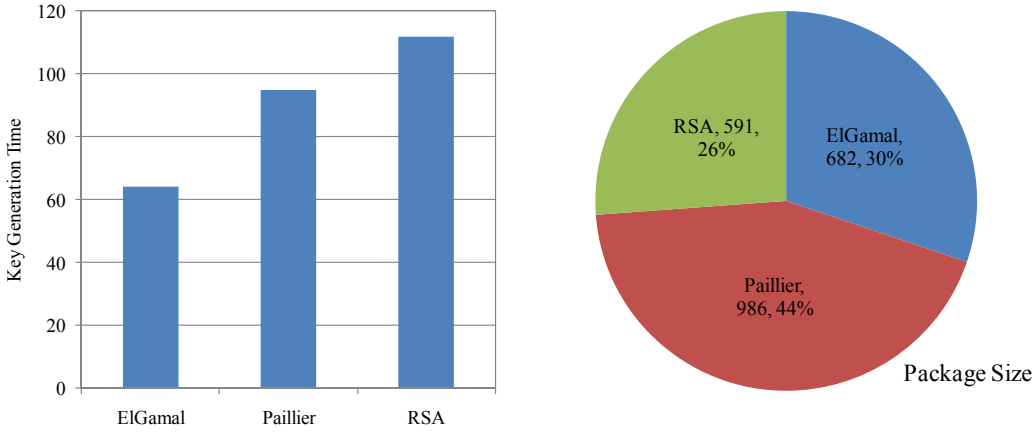


Figure 5.5 Key generation time and package size

In case of the package size, because the encryption of the RSA is only one number, so the posted package is smaller that ElGamal and Paillier. Both of these methods encrypt the one plaintext to the two ciphertexts. The Paillier packages are the biggest one due to the number of parameters which are used in Paillier system to encrypt a plaintext.

Base on the result, even though the RSA package is smaller than ElGamal package, but regarding to the speed and easiest key management of the ElGamal, it seems that the ElGamal is the best choose for implementing in online applications.

5.2.2 E-Voting Experimental Result Using the Online Secret Sharing

Suppose that, there is *n* voter and they have their own private votes. The voters can join to the election process any time they want. The Teller system should

calculate the total of respondents' votes. Obviously the teller is not able to compute the exact vote of the respondents. The main parameters of experiment are: Number of Voters and the range of total votes *V*.

The algorithm's main stages are same as previous experiment, except that the *X, Y* should compute in Teller system instead of Precomputing offline. The stages are:

- Sending Encrypted message to Teller
- Computing *m, h, X, Y* on Teller
- Finding desired d across its' possible values

In first phase, each voter owns a Boolean data. The final result of phase 1 is shown Table 5.2.

Table 5.2 Processing Time of Online Secret Sharing Algorithm

Number of Voters	**Send Message**	**Computing *m, h, X, Y* on Teller**	**Finding Desired *V***	**Total**
100	28	21.83	124.08	173.69
500	33	110.73	620.01	763.58
1,000	39	240.73	1240.20	1520.11
5,000	90	1179.08	6200.62	7469.57
10,000	153	3013.71	12401.34	15568.26
50,000	660	12180.72	62006.40	74847.12
100,000	1294	22060.96	124017.79	147372.25
500,000	6361	139185.28	620003.12	765549.86
1,000,000	12696	375163.02	1240169.67	1628029.10

In this case there are two options for voters. They can vote their desired item or reject it. In sooth, the algorithm counts the users' votes not sum them. The Teller aims to know how many users do vote the idea. In this experiment, the respondents send the Boolean data to Teller. In this condition, the range of *V* is equal to the

number of voters. A range between 100 and 1,000,000 users are contributed in this experiment and the earned time is base on the average of five algorithm runs. The time is highly related to the number of users. For example, Teller computation takes 27 minutes for 1 million voters. The result is also shown in Figure 5.6.

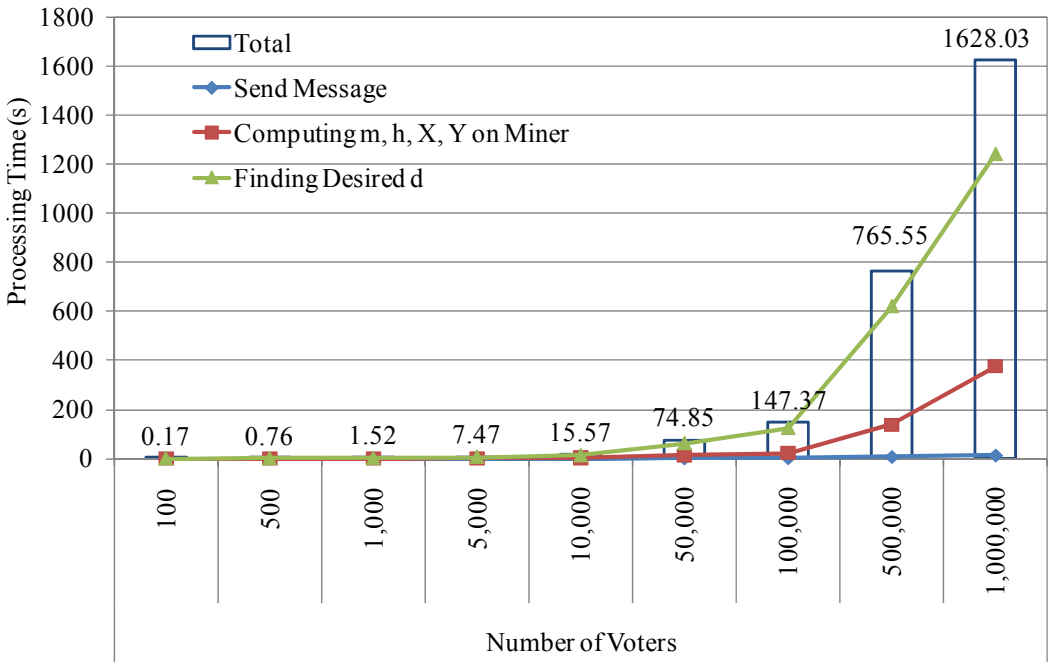


Figure 5.6 Total time of Algorithm execution related to the Number of respondents

Second phase is dedicated to non Boolean cases. Suppose that a web administrator wants to know his visitors' opinion about a subject. He asks them about site attractive and wants them to vote from weak, not bad, good, attractive and high attractive. Each option is weighted from 1 to 5. The aim of administrator is to calculate the average points of visitors' opinion. In fact the web poll should sum all votes and then divide it to the number of visitors who contribute in pool. On the other hand, the visitors do not like to reveal their vote to the administrator for privacy purpose. In this case, each user sends an integer value instead of the Boolean value and the range of V is different from the number of users. For example, in mentioned sample the range of V is 5 × Number of users. So there are two main parameters that should be considered: Number of the visitors and the Range of V. In this case, the time is highly related to the range of V. The algorithm is tested over a range of 100 to 1 million users and a range of V between 100 to 100 million.

The result is shown in Figure 5.7.

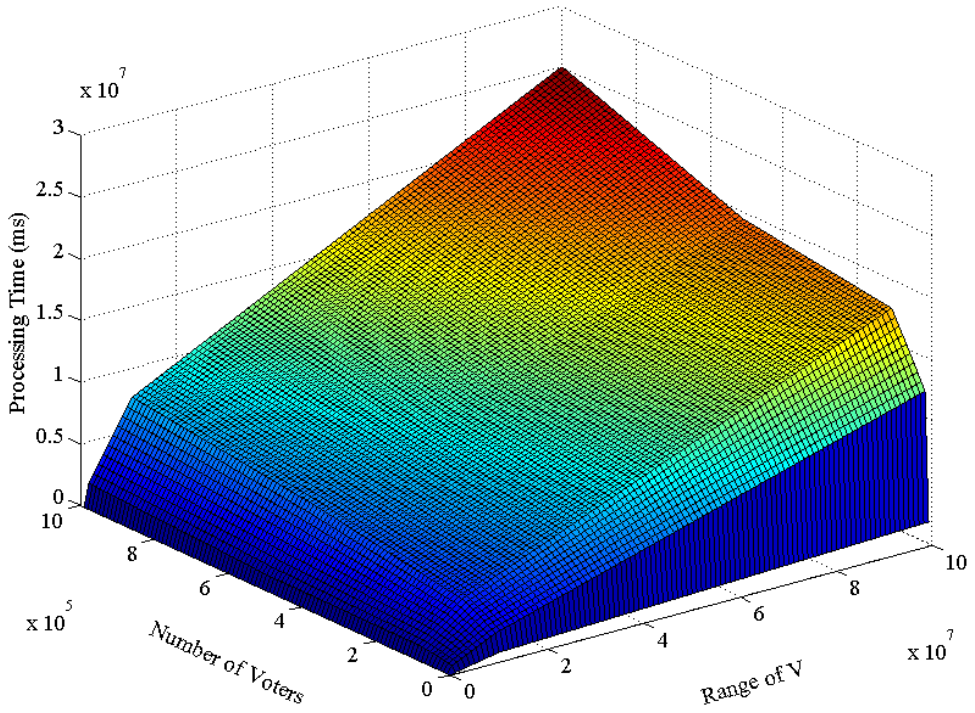


Figure 5.7 Total time related to Number of Voters and Range of *V* – 3d view

Although, the processing time is highly related to the number of visitors, but the range of *V* should not be ignored. The result shows that by increasing the range of *V* the final processing time extremely increased. It is due to the complexity of the Teller algorithm's loop; because the loop variable is *V*. If the number of user increases, the time of computing *m, h and* shared keys will increase. Note that two shared secret keys Ω, Ψ are also furthered. Even though, this mutation is considerable, but the range of *V* and its role in try and error loop is more significant. The algorithm lasts less than 40 minutes for 1 million voters with *V* in the range of 1 to 100 million.

5.2.3 Online Consecutive E-Voting; Experimental Result

The third experiment is involved on implementing an e-voting process using the online consecutive multi party computation algorithm as shown in Figure 5.8.

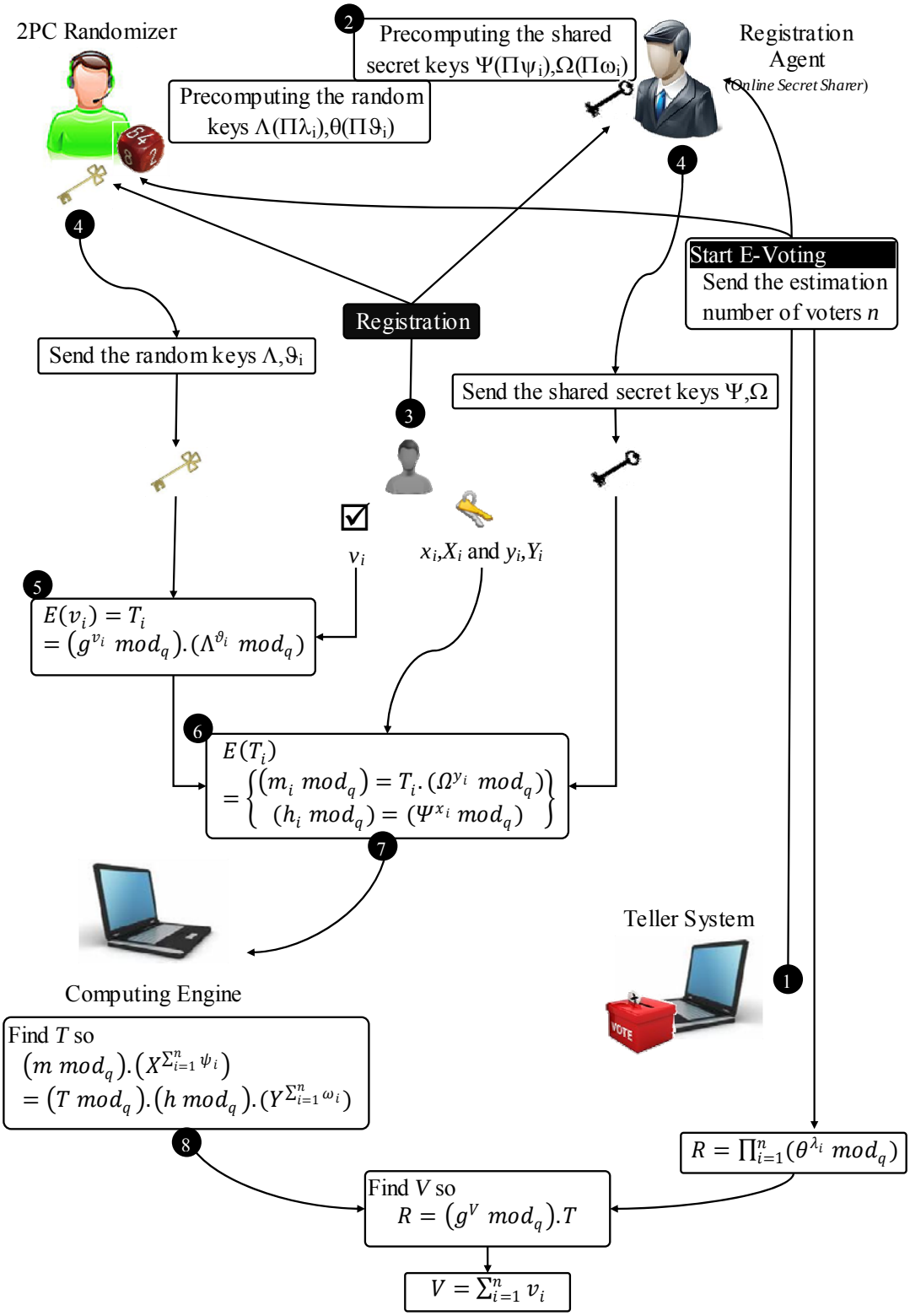


Figure 5.8 Online E-Voting process

In this experiment, two components are added to the system architecture; the 2PC randomizer and the computing engine. Surely, the overload of computations in these new components increase the process time and the response time of the e-

voting result. Since, those components are independent, so some parallel computations can run simultaneously.

The main stages which affect the response time directly are;

- Sending the Encrypted message to Teller
- Computing the m, h, X, Y by the computing engine.
- Finding the temporary random result T by the computing engine.
- Finding desired V across its' possible values by the Teller.

Note that there are some computations that can precompute offline or parallel, so those are ignored in the total process time. For example, preparing the set of the random keys by 2PC randomizer or the shared secret keys by register agent can precompute before the process starts. On the other hand, calculating the encrypted vote by voters and sending the shared secret and random keys to the voters can compute in parallel. There are some transferring data that happen only one time such as sending the derandomize parameter R by the 2PC randomizer to the aggregator, or sending the temporary random result by the computing engine to the aggregator which can be ignored when the number of users increase.

The result of the online consecutive e-voting algorithm is shown in Table 5.3 and Figure 5.9, respectively. The result is obtained for the range of the 100 to 1,000,000 voters. The response times are based on average of 5 runs. According to the result, unlike the previous phases the time is absolutely unrelated to the number of voters. Although, the range of V affects the response time, but the influence of computing the V by the aggregator is insignificant. The most important factor is the range of the random keys, which is chosen by the 2PC randomizer.

Table 5.3 Processing Time of Online Consecutive Algorithm

Number of Voters	Send Message	Computing *m*, *h*, *X*, *Y* by Teller	Computing the temporary result *T*	Finding the votes' summation *V*	Total
100	28	19.83	9581464.00	121.08	9581632.60
500	33	105.26	9692303.00	605.24	9693046.39
1,000	39	256.32	7913787.00	1251.32	7915334.03
5,000	91	1212.36	6885969.00	6319.54	6893592.29
10,000	156	3111.50	6952785.00	12326.64	6968379.53
50,000	676	12254.32	6492337.00	61886.86	6567154.57
100,000	1326	21589.64	8660464.00	125124.30	8808504.33
500,000	6526	138549.66	7200263.00	619876.40	7965215.45
1,000,000	13026	381452.34	8231348.00	1238986.20	9864812.93

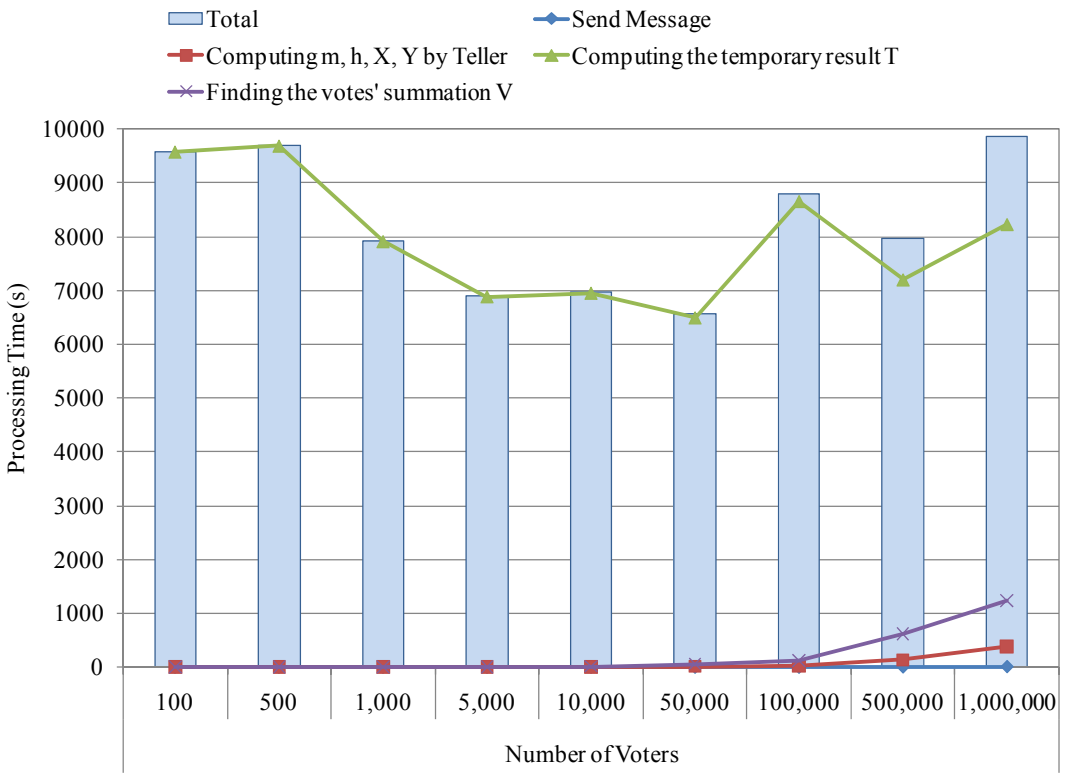


Figure 5.9 The online consecutive processing time related to number of voters

However, the size of the random keys assesses by 2PC randomizer, but the

most time consuming step is the computation of the temporary random result *T* which is calculated by the computing engine. Note that this time is independent of the number of voters or even the range of the votes' summation *V*. In the experiment, 64 bits random keys are employed by the simulated 2PC randomizer. In case of the 500,000 and 1,000,000 voters, however the time of the finding result *V* in the aggregator is highly increased, but it is much less than the time consumed for the calculation of the temporary random result. Base on the result, the total time approximately is equal to the computing time that is consumed for calculation of the temporary random result *T*. This interpretation means that the implementation should focus on the speed of the computing engine. High speed computing engine leads to low response time directly. The percentage portion of each step among the total processing time is shown in Figure 5.10.

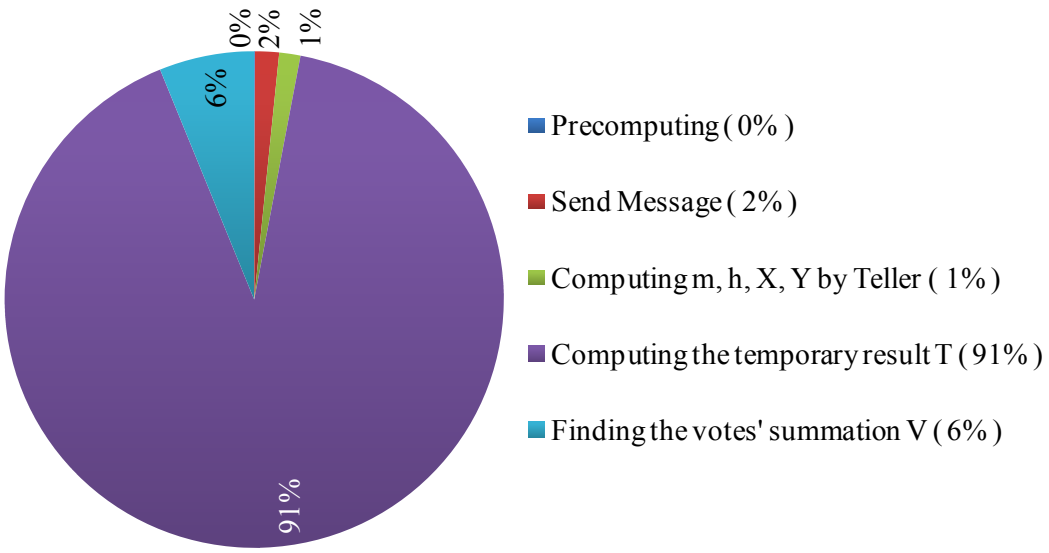


Figure 5.10 The percentage portion of each step among the total processing time

Due to the size of the random keys, finding the temporary result is the most time consuming process that is around 91% of total time. The small size of the random keys absolutely decreases the computation time, but the security of the randomness significantly decreased. The 1,000,000 voters can wait around two hours for the result, but they do not join to the low security e-voting. Since, the bottleneck is the computing engine, using the high speed multi processor will decrease the response time significantly.

5.3 Distributed Frequent Itemset Mining Experimental Result

Same as before, The IIS HTTP Server is used for network simulation under windows Vista on a PC with a 2.4GHz processor and 2GB memory. The 512 bits are chosen as the length of each cryptographic key. All databases are implemented in SQL-Server 2005. The experiments are arranged in two different phases according to the two important factors which are affecting highly in the algorithm's response time. These two important parameters are;

1. The number of transactions *n*.
2. The number of parties *N*.

In the first phases, the number of parties is fixed and in every experiment, number of the transactions is increased and the execution time is investigated. In this experiment, number of the parties is five and each party owns the seven fields averagely. The experiment is conducted over 100 to 10,000,000 transactions respectively. The execution time is shown in Table 5.5.

Table 5.4 Execution time in millisecond related to the number of transactions

Number of Transactions	**100**	**1000**	**10000**	**100000**	**1000000**	**10000000**
Sort Transactions	0.40	7.97	66.44	1494.87	15945.25	186027.97
Creating Boolean Support Matrix	0.02	0.17	1.25	20.00	108.12	1006.64
Creating Boolean Support Vector	0.01	0.14	1.05	9.98	102.24	1124.32
Send Vector	33	91	676	5526	24012	140524
Computing Multi Party Aggregation	1.00	10.96	186.03	1827.06	27904.20	325548.95
Frequent Decision	0.02	0.04	0.14	1.43	12.50	123.48
Total	34.44	110.28	930.91	8879.34	68084.31	654355.37

There are some stages that are related to the number of transactions, which are listed in the result table. Some of the stages are significantly dependent to the database engine such as sorting the transactions. It is noted that all the databases are designed in SQL-Server 2005 and the speed of the database engine effects directly to the algorithm's response time. All steps which are focused in the experiment are listed below as shown in Table 5.5.

- Sort Transactions
- Creating Boolean Support Matrix
- Creating Boolean support Vector
- Send Vector
- Computing Multi Party Aggregation
- Frequent Decision

As mentioned before, the first step named as sorting transactions, is quite related to the database management systems and its selected algorithm for sorting that will not be discussed here. Briefly, a simple interpolation and curve fitting over the data shows that the sorting algorithm in SQL-Server 2005 behaves logarithmic related to the number of sorting transactions. The second step, computing the Boolean support matrix is a simple computation over the transaction table. Surely, the computing time is related to the number of matrix's elements equals to the width multiplication by the height of the transaction database. Anyway, by increasing the number of transactions, the computing Boolean support matrix time is much less than overall time so that it is possible to be ignored. The third step, behaves exactly same as the second step. Although, the number of the transactions increases the time of computing the Boolean support vector, but the mentioned time is not much enough big to effect on overall time.

The first effective parameter is sending the support vector. The send time is quite associated with the network specification. Each party should send the support

vector element by element separately, which equals to the number of transaction. On the other hand, number of all elements is the number of transactions multiplication by the number of parties. So the send time is related to both number of the transactions and the number of businesses who join to the mining process. Since the number of companies is fixed in this experiment, so the execution time is only based on the number of transactions. It is obvious that increasing the number of transactions raises the execution time.

Since the miner should aggregate all parties' data to discover that a transaction supports the itemset or not, so the miner should compute the multi party aggregation for each transaction. In fact, the number of multi party aggregation that is computed by the miner is the cardinality of the support vector, which equals to the number of transaction. The computing of the multi party aggregation is also time consuming process because of its dependency to the number of transactions.

The overall execution time is depicted in Figure 5.13.

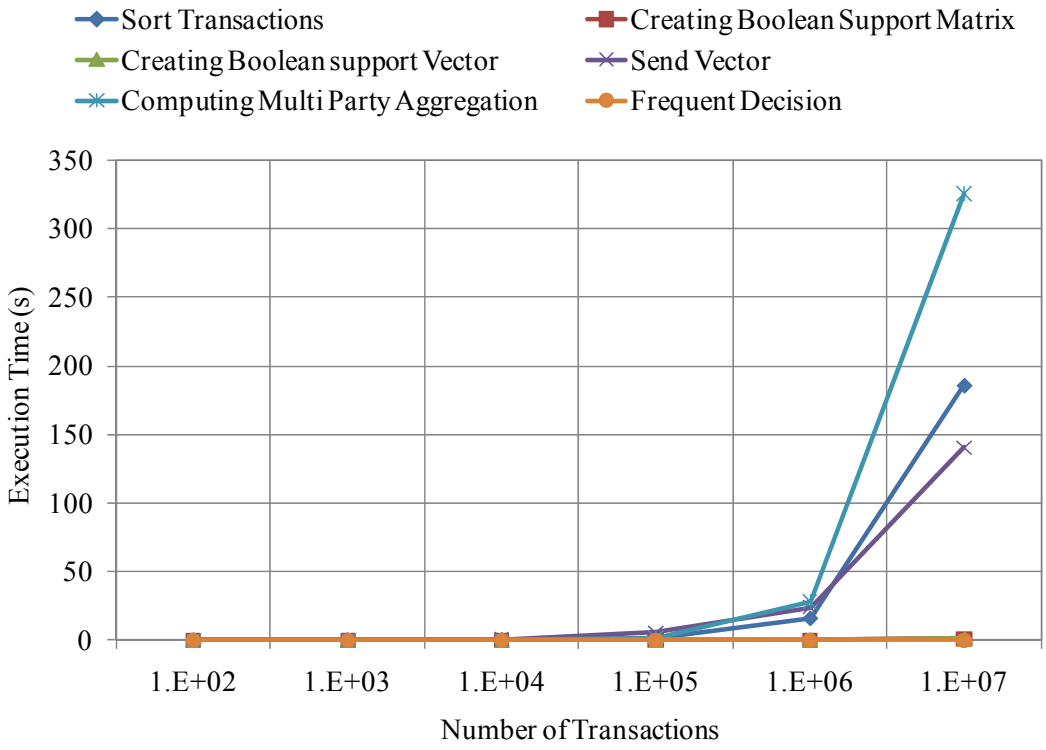


Figure 5.11 Execution time related to the number of transactions

The last step is calculating the summation of all supported transactions by miner to decide whether the itemset is frequent or not. The simple summation by the miner does not consume much time, which can affect to overall time significantly.

In the second phase of the experiment, the number of transaction is supposed to be fixed and the number of parties is changed. In this experiment, all transaction databases include 1,000,000 transactions and the number of businesses who join the mining process are varying between 3 and 20 companies. Two steps are focused in this experiment due to their relation with the number of parties. These two steps are;

- Computing Multi Party Aggregation
- Send Vector

The computing multi party aggregation; as its name implies; seems to be related to the number of the parties. Based on the result shown in Figure 5.14, contrary to the supposition, increasing the number of parties does not impact on the execution time. The time does not change greatly, because the miner does not need to compute the exact value of support vector aggregation, the miner only needs to compute that the summation equals to the number of parties N or not. Most time consuming stage of the multi party computation is the trial and error step that the miner tries to find the exact value of summation;

```
For All Possible (d) do
      If R = (g^d mod_q).T then
            return (d);
```

In this case, the miner does not need to test all possible d to find the exact value of the aggregation result. In fact, the miner only tests that the aggregation result equals to N or not. It is noted that, if the aggregation result equals to the number of parties then the transaction supports the itemset, otherwise the transaction does not support the itemset. This fact is summarized below;

```
If R = (g^N mod_q).T then
        Y_k = 1 // The k^th transaction supports itemset
Else
        Y_k = 1 // does not supports itemset
```

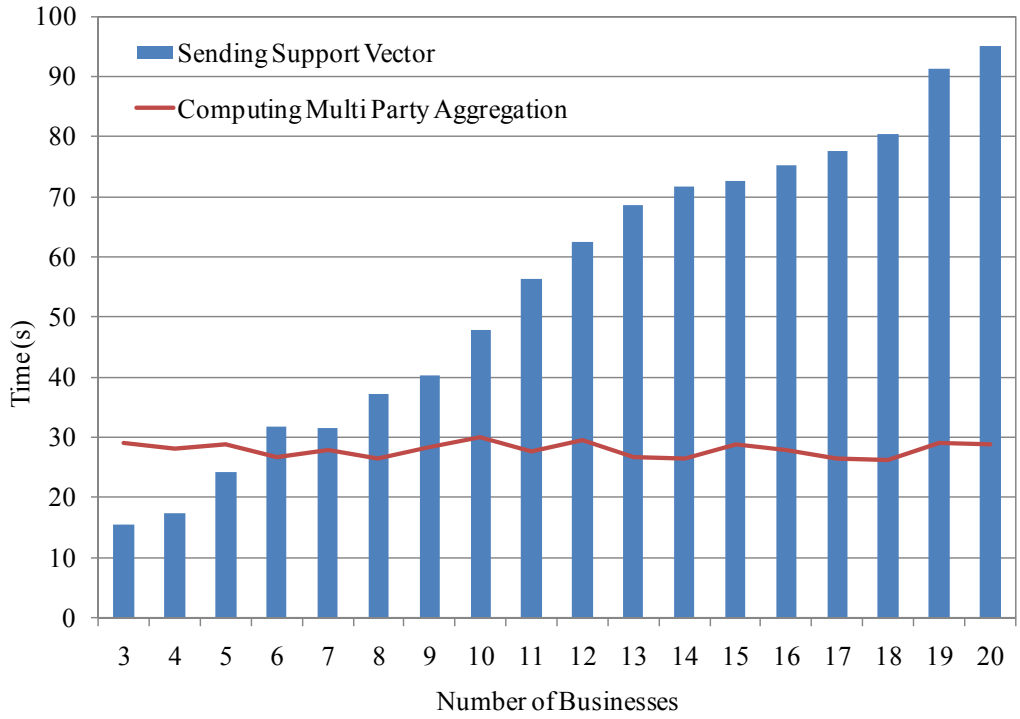


Figure 5.12 Execution time related to the number of parties

On the other hand, as mentioned before, the send time increase linear with growth of the parties' number. Each party sends his own support vector. The support vector includes *n* numbers, where *n* is the number of transactions. Thereupon, total number of sends equals to the number of parties multiplication by the number of transactions.

Some network considerations such as peer to peer connection between the parties and the miner (the aggregator or even the computing engine) for parallel sending can decrease the overall send time.

5.4 Summary

In this chapter, the designed online consecutive secure multi party computation algorithm is testes to prove the performance of the algorithm in real world application. The experiment is performed on windows vista with a 2.4GHz processor, SQL-Server and IIS Server for network simulator.

The experiment is divided to two different application; online e-voting and distributed frequent itemset mining. In the first experiment, a classic example known as e-voting is developed to test the performance of the designed algorithm. Three phases according to the three versions of the designed algorithm are investigated ad discussed. Boolean votes simulate the electing process and non Boolean votes imitate the web polls. E-voting established with the enhanced offline ElGamal based multi party computation shows a low speed response time. Although, the web polls using online secret sharing lasts 27 minutes for one million voters, but the process is online all the process life. In the last phase, a full online e-voting is developed. Due to the 2PC randomizing, the process time is increased, but the confidentiality of the votes is guaranteed.

The second experiment is developing the vertically distributed frequent itemset mining using the designed algorithm. The mining model and algorithm are fully described. The experiments show that three steps are affecting the overall response time. The first one is sorting transactions, which are highly related to the database management system. The second step is the sending support vector that is exactly associated with the network structure. The last one is the computing multi party aggregation that is based on the number of the transactions. Because of the form of the computation, the step is independent of the number of the parties. The result ensures the performance and applicability of the designed algorithm in real world application.

CHAPTER 6

CONCLUSION

6.1 Introduction

In this paper, a secure simultaneous aggregation algorithm is developed. The proposed algorithm ensures the confidentially of sensitive respondents' data. Furthermore, it does not need communication channels between different respondents or multi round interaction between any respondent and the Miner Systems. Most multi party computation algorithms share a secret between all users to perform a secure computation. All users should be present at the start time of the process to miner shares secret between them. But in most real problem especially web based problem, the users can join the system every time they want. The proposed method is able to compute the aggregation of the respondents' data securely and simultaneously.

The mathematical demonstration ensures the accuracy of data's aggregation. On the other hand, The DDH assumption and the ElGamal cryptosystem ensure the privacy of the algorithm and respondents' data, so the Miner system can not reveal respondents' data. Our experimental results over a real E-Voting system also show significantly desirable response time. The time increasing is derived from high security of algorithm, because the Miner could not able to decrypt the message, so the Miner should use trial and error to find the result.

The algorithm can also be used for any model enabled by counting values. Our both theoretical analysis and proof in addition to experimental results show that

the algorithm is very efficient and runs in desirable time. The following sections evaluate the new segmentation operator in relation to other segmentation methods.

6.2 Evaluation of the Designed Algorithm

Mainspring of this research is designing an online multi party computation algorithm based on current offline algorithms and overcome to the progressive problem. Since the base of the research is the current offline algorithm, the first step of designing the online algorithm is enhancing the existing algorithm to show better performance in online applications. ElGamal cryptosystem is selected as the fundamental element of a multi party computation algorithm to improve the features of the algorithm, which are important in the online environment. The most important characteristic of ElGamal cryptosystem, which makes it suitable for this research, are; speed, easy key management, exponential ability and low computation complexity. An ElGamal based offline multi party computation algorithm is developed in the first step and fully discussed. In this model, there are some participants who join together and use their ElGamal crypto keys to make an offline shared secret. Then each participant uses his ElGamal private keys and the agreed shared secret keys to send the data to the aggregator, securely. The algorithm security has been proven mathematically and the confidentially of the participants' data has been guaranteed.

The studies conducted on the current algorithms shows that the dependency between the participants' keys and the shared secret key leads to an offline secret sharing process. In the second step, the infinite convergent product sequences have been employed as the shared secret key to overcome to the dependency problem. Using the infinite convergent product series has two main benefits; first the shard secret key is independent of the participants' keys so the process can continue online during application life and second the infinite feature of the product sequences provides the possibility of the infinite number of participants who like to join the process. In this step, an online secret sharer has been designed using the infinite convergent product sequences. The online secret sharer prepares two infinite series

as the shared secret keys and shares them among the participants who will join the process. In this model, the parties can participate in the process all the process life and any time they wish. Unlike the offline secret sharing, the parties do not need to meet each others to make the shared secret before process starts.

When the multi party computation algorithm runs online, an important problem will occur. Suppose that the participants can join the process any time they wish, it means that the aggregator can compute the aggregation result any time he wants. This assumption leads to a big security leak which gives this ability to the aggregator to compute the participants' data without decrypting or any complex calculation. Since the aggregation result is an exact number not a comparison, the aggregator can compute the result at the time i, i+1 in tandem and simply subtract them to earn the (i+1)th party's data. The problem is essentially same as the two party computation problem. In two party computation problem, it is impossible to compute the summation of two millionaires' wealth, so that they cannot calculate the other's wealth. It is obvious that while a millionaire knows the summation and due to his knowledge about his wealth, he can compute the next millionaire's wealth. The online consecutive model has been designed to solve the two party computation problem and ensures the 2PC honest model. In the 2PC honest model at least three components should exist to make the model honest and secure. In the purposed algorithm, a 2PC randomizer has been added to the model to randomize the participants' data for preserving the confidentially of information. Due to the randomization process, the computing engine cannot compute the parties' data by the consecutive calculation. The computing engine can only compute the temporary randomized result. It is too important to separate the 2PC randomizer and the computing engine and disconnect the entire communication channel between themselves. The aggregator finally computes the exact result using the derandomization parameters posted by the 2PC randomizer. The 2PC honest and semi honest models ensure the confidentially of the participants' sensitive data.

In addition, two case studies have been studied to test the performance and especially applicability of the designed algorithm. In the first case study, the classic example known as e-voting has been investigated to ensure the performance of the

designed algorithm. The execution time of the algorithm has been selected for the target parameter of the experiment. The result has been divided to the three different phases according to the three stages of the designed algorithm. First the ElGamal based multi party computation has been tested over the yes/no votes. The result shows that the algorithm lasts only 0.5 second for thousand of the voters. In the next experiment, the integer votes have been trained. The integer votes denote the web polls with more than two choices for the voters. In this case, two parameters are affecting to the response time; number of the voters and the range of the choices. The result shows the 70 seconds for thousand voters with utmost one million for the result's range. According to the designing procedure, in the second phase, the e-voting has been employed the online secret sharer to establish an online e-voting. It is obvious that the time will increase due to computations related to the infinite convergent product series. The result shows the 27 minutes in case of the yes/no votes with millions of voters and the 40 minutes in case of the web polls for millions of voters and utmost 100 million for the result's range. The experiments are satisfactory related to the number of voters. In the last phase, a complete model according to the designed online consecutive multi party computation algorithm has been formed and tested over the variant range of the data. The e-voting process has been implemented using the 2PC randomizer, the computing engine and the aggregator to ensure the 2PC honest model. Based on the result, the mist time consuming step is the computing the temporary randomized result which allocates 90% of the overall time.

Although the application of the multi party computation is disregarded by the recent researchers, the distributed frequent itemset mining had been developed using the designed algorithm to ensure the applicability of the algorithm. In this model, there are some businesses, which own the vertically distributed data among their transaction databases. They interest to join the mining process over all the databases and benefit from the discovered rules and patterns. Deterrent factor is privacy of the confidential information related to the clients, patients and people. In the purposed distributed frequent itemset mining algorithm, all businesses use the secure multi party computation algorithm to send their portion of itemset's support vector. So the secure multi party computation algorithm guarantees the confidentially of the data and each company ensures that no more information reveal except the mining result.

The model and algorithm have been fully discussed and tested based on the number of transactions and number of the businesses which are most important factors associated with the application. Three most time consuming steps are sorting the transactions, sending the messages and computing the multi party aggregation. The first step is significantly related to the selected database management system which was SQL-Server 2005 in the experiment. The second step varies base on the established network and communication channel and servers which are used for connecting the components. Regarding to the result, the computing multi party aggregation's time only increases if the number of transaction enlarges. The time is quite independent of the number of businesses, because the aggregator only needs to compute the summation of support vector equals to the number of businesses or not, despite of what is the number of businesses. The result emerges the potentiality of the designed algorithm to solve the real world online application, especially in attractive data mining areas.

Next, this research contribution is briefly enumerated.

6.3 Thesis Contribution

The main contributions performed in this research are:

1. Develop an ElGamal based offline multi party computation algorithm with high-speed and easier to use.
2. Creating the online secret sharing process using convergent infinite product series to produce the online multi party computation algorithm.
3. Improving the online aggregation algorithm using two party computation techniques and dominate the sensitive information revealing problem.
4. Applying the innovative algorithm to data mining tools and ensure the applicability of the algorithm.

6.4 Further Work

Sea of the knowledge is never ending. More to be explored as suggested below;

- Continue to do more on online secret sharing process and techniques.
- Detailing more on the time bottleneck of the algorithm to gain the faster response time.
- Investigating the external, network and crypto attacks.
- Inspecting the network requirement, server specification and infrastructures which are needed to implement in the real world environments.

REFERENCES

Agrawal, D. and Agrawal, C. (2001). On the design and quantification of privacy preserving data mining algorithms. *Proceedings of 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems***,** pp 247–255.

Agrawal, D. and Srikant, R. (2000). Privacy preserving mining. *Proceedings of the 2000 ACM SIGMOD Conference on Management of Data*. pp 439–450.

Agrawal, R., Imielinski, T. and Swami, A. (1993). Mining Association Rules Between Sets of Items in Large Databases. *SIGMOD Conference***,** pp 207-216.

Agrawal, R. and Srikant, R. (2007). Fast algorithms for mining association rules in large databases. *Proceedings of the 33th International Conference on Very Large Data Bases***,** pages 487-499.

Aiello, W., Ishai, Y. and Reingold, O. (2001). Priced oblivious transfer: how to sell digital goods. *EUROCRYPT* 119-135.

Amirbekyan, A. and Estivil-Castro, V. (2007). Privacy-Preserving Regression Algorithms. *Proceedings of the 7th WSEAS International Conference on Simulation, Modelling and Optimization*, Beijing, China.

Androulaki, E. and Bellovin, S. (2010). A Secure and Privacy-Preserving Targeted Ad-System. SION, R., CURTMOLA, R., DIETRICH, S., KIAYIAS, A., MIRET, J., SAKO, K. and SEBÉ, F. (eds.). *Financial Cryptography and Data Security***.** Springer Berlin / Heidelberg. 6054, 123-135.

Assaf, B.-D., Noam, N. and Benny, P. (2008). FairplayMP: a system for secure multi-party computation. *Proceedings of the 15th ACM conference on Computer and communications security*, Alexandria, Virginia, USA. ACM.

Atallah, M. J. and Du, W. (2001). Privacy-preserving cooperative scientific computations. *14th IEEE Computer Security Foundations Workshop*. 273–282.

Atallah, M. J., Du, W., Dehne, F. K. H. A., Sack, J. R. and Tamassia, R. (2001). Secure multi-party computational geometry. *Lecture Notes in Computer Science, Springer,* 2125**,** 165-179.

Bagüés, S. A., Zeidler, A., Matias, I. R., Klein, C. and Valdivielso, C. F. (2010). Enabling Personal Privacy for Pervasive Computing Environments. *Journal of Universal Computer Science,* 16**,** 341-371.

Baudron, O., Fouque, P. A., Pointcheval, D., Poupard, G. and Stern, J. (2001). Practical Multi-Candidate Election System. *Twentieth Annual ACM Symposium on Principles of Distributed Computing (PODC'01).* ACM Press, 274-283.

Beaver, D. (1997). Commodity-based cryptography. *twenty-ninth annual ACM symposium on Theory of computing*. 446-455.

Beaver, D. and Goldwasser, S. (1989). Multiparty computation with faulty majority. *Advances in cryptology CRYPTO '89*, New York, NY, USA. Springer-Verlag, 589-590.

Bella, G. (2008). What is Correctness of Security Protocols? *Journal of Universal Computer Science,* 14**,** 2083--2107.

Bertino, E., Fovino, I. N., Parasiliti, L. and Provenza (2005). A Framework for Evaluating Privacy Preserving Data Mining Algorithms*. *Data Min. Knowl. Discov.,* 11**,** 121-154.

Blosser, G. and Zhan, J. (2008). Privacy-Preserving Collaborative E-Voting. *Proceedings of the IEEE ISI 2008 PAISI, PACCF, and SOCO international workshops on Intelligence and Security Informatics*, Taipei, Taiwan. 1425588: Springer-Verlag, 508-513.

Blundo, C. and Masucci, B. (1999). Randomness in Multi-Secret Sharing Schemes. *Journal of Universal Computer Science,* 5**,** 367--389.

Bogetoft, P., Damgård, I., Jakobsen, T., Nielsen, K., Pagter, J. and Toft, T. (2006). A Practical Implementation of Secure Auctions Based on Multiparty Integer Computation. *Financial Cryptography and Data Security.* 142-147.

Boneh, D. (1998). The decision Diffie-Hellman problem. *Lecture Notes in Computer Science*.

Brassard, G., Cr´epeau, C. and Robert, J. M. (1987). All-or-nothing disclosure of secrets. *Advances in cryptology-CRYPTO '86*, London, UK. Springer-Verlag, 234-238.

Brickell, J. and Shmatikov, V. (2005). Privacy-preserving graph algorithms in the semi-honest model. *ASIACRYPT***,** 236-252.

Cachin, C. (1999). Efficient private bidding and auctions with an oblivious third party. *6th ACM conference on Computer and communications security*. 120-127.

Camenisch, J. and Shoup, V. (2003). Practical verifiable encryption and decryption of discrete logarithms. *CRYPTO 2003. LNCS, Springer, Heidelberg,* 2729**,** 126-144.

Canetti, R. (2000). Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology,* 13**,** 143-202.

Canetti, R., Lindell, Y., Ostrovsky, R. and Sahai, A. (2002). Universally composable two-party and multi-party secure computation. *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, Montreal, Quebec, Canada. 509980: ACM, 494-503.

Chaum, D., Crpeau, C. and Ivan, D. (1988). Multiparty unconditionally secure protocols. *Proceedings of the twentieth annual ACM symposium on Theory of computing*, Chicago, Illinois, United States. ACM.

Chen, K. and Liu, L. (2010). Geometric data perturbation for privacy preserving outsourced data mining. *Knowledge and Information Systems***,** 1-39.

Chen, L., Susilo, W., Wang, H., Wong, D. S., Dawson, E., Lai, X., Mambo, M., Miyaji, A., Mu, Y., Pointcheval, D., Preneel, B. and Smart, N. (2008). Cryptography in Computer System Security. *Journal of Universal Computer Science,* 14.

Chiew, K. (2008). Data Mining with Privacy Preserving in Industrial Systems. LIU, Y., SUN, A., LOH, H., LU, W. and LIM, E.-P. (eds.). *Advances of Computational Intelligence in Industrial Systems.* Springer Berlin / Heidelberg. 116, 57-79.

Choi, S., Elbaz, A., Juels, A., Malkin, T. and Yung, M. (2007). Two-Party Computing with Encrypted Data. KUROSAWA, K. (ed.) *Advances in Cryptology – ASIACRYPT 2007.* Springer Berlin / Heidelberg. 4833, 298-314.

Chor, B., Goldwasser, S., Micali, S. and Awerbuch, B. (1985). Verifiable secret sharing and achieving simultaneity in the presence of faults. 26th IEEE Symposium on Foundations of Computer Science. 383-395.

Clifton, C., Kantarcioglu, M. and Vaidya, J. (2002). Defining Privacy for Data Mining. *National Science Foundation Workshop on Next Generation Data Mining*.

Cramer, R., Damgard, I., Dziembowski, S., Hirt, M. and Rabin, T. (1999). Efficient Multi-party Computations Secure Against an Adaptive Adversary. *Advances in Crytology-EUROCRYPT '99,* 1592 of Lecture Notes in Computer Science (LNCS)**,** 311-326.

Cramer, R., Gennaro, R. and Schoenmakers, B. (1997). A Secure and Optimally Efficient Multi-Authority Election Scheme. FUMY, W. (ed.) *Advances in Cryptology — EUROCRYPT '97.* Springer Berlin / Heidelberg. 1233, 103-118.

Dahlia, M., Noam, N., Benny, P. and Yaron, S. (2004). Fairplay;a secure two-party computation system. *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, San Diego, CA. USENIX Association.

Damg˚ard, I. and Koprowski, M. (2001). Practical Threshold RSA Signatures Without a Trusted Dealer. *Advances in Cryptology - EUROCRYPT 2001, LNCS, Springer Verlag,* 2045**,** 152-165.

Damgård, I. and Jurik, M. (2001). A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography*. 746742: Springer-Verlag, 119-136.

Damgård, I. and Nielsen, J. (2003). Universally Composable Efficient Multiparty Computation from Threshold Homomorphic Encryption. *In Advances in Cryptology - CRYPTO'03*. 2729, 247-264.

Damgård, I. and Thorbek, R. (2007). Non-interactive Proofs for Integer Multiplication. NAOR, M. (ed.) *Advances in Cryptology - EUROCRYPT 2007.* Springer Berlin / Heidelberg. 4515, 412-429.

Das, A. S., Keshri, J. K., Srinathan, K. and Srivastava, V. (2008). Privacy preserving shortest path computation in presence of convex polygonal obstacles. *Third International Conference on Availability, Reliability and Security (ARES '08)*, Washington, DC, USA. IEEE Computer Society, 446–451.

Dasseni, E., Verykios, V. S., Elmagarmid, A. K. and Bertino, E. (2001). Hiding association rules by using confidence and support. *Information Hiding Workshop***,** pp 369–383.

Desmedt, Y. and Frankel, Y. (1990). Threshold Cryptosystems. *Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*. 705196: Springer-Verlag, 307-315.

Desmedt, Y. and Frankel, Y. (1994). Perfect Homomorphic Zero-Knowledge Threshold Schemes over any Finite Abelian Group. *SIAM J. Discrete Math,* 7**,** 667-679.

Diffie, W. and Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory,* IT-22**,** 644-654.

Du, W. and Atallah, M. J. (2001). Secure multi-party computation problems and their applications: a review and open problems. *Proceedings of the 2001 workshop on New security paradigms*, Cloudcroft, New Mexico. 508174: ACM, 13-22.

Du, W. and Atallah, M. J. (2002). Privacy-preserving cooperative statistical analysis. *17th Annual Computer Security Applications Conference*. 102–110.

Du, W., Han, Y. S. and Chen, S. (2004). Privacy-Preserving Multivariate Statistical Analysis: Linear Regression and Classification. *Fourth SIAM International Conference on Data Mining*. 222-233.

Du, W. and Zhan, Z. (2002). A practical approach to solve Secure Multi-party Computation problems. *workshop on New security paradigms*. 127-135.

ElGamal, T. (1985). A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory,* 31 469-472.

Estivill-Castro, V. and Brankovic, L. (1999). Data Swapping: Balancing Privacy against Precision in Mining for Logic Rules. *Lectures in Computer Science*.

Even, S., Goldreich, O. and Lempel, A. (1985). A randomized protocol for signing contracts. *Communication ACM,* 28**,** 637-647.

Evfimievski, A., Srikant, R., Agrawal, D. and Gehrke, J. (2002). Privacy preserving mining of association rules. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining***,** pp 217–228.

Farras, O. and Padro, C. (2009). Ideal Hierarchical Secret Sharing Schemes. *Cryptology ePrint Archive*.

Fouque, P. A., Poupard, G. and Stern, J. (2001). Sharing Decryption in the Context of Voting or Lotteries,. *Financial Cryptography , LNCS, Springer Verlag,* 1962**,** 90-104

Franklin, M. K. and Reiter, M. K. (1997). Fair exchange with a semi-trusted third party. *4th ACM conference on Computer and communications security*. 1-5.

Frikken, K. B. (2007). Privacy-preserving set union. *ACNS*, 237-252.

Frikken, K. B. and Atallah, M. J. (2003). Privacy preserving electronic surveillance. *2003 ACM workshop on Privacy in the electronic society*, New York, NY, USA. ACM, 45–52.

Furukawa, J., Miyauchi, H., Mori, K., Obana, S. and Sako, K. (2009). An Implementation of a Universally Verifiable Electronic Voting Scheme Based on Shuffling. BLAZE, M. (ed.) *Financial Cryptography*. Springer Berlin / Heidelberg. 2357, 16-30.

Ge, W., Wang, W., Li, X. and Shi, B. (2005). A Privacy-Preserving Classification Mining AlgorithmThis paper was supported by the National Natural Science Foundation of China (No.69933010, 60303008) and China National 863 High-Tech Projects (No.2002AA4Z3430). HO, T. B., CHEUNG, D. and LIU, H. (eds.). *Advances in Knowledge Discovery and Data Mining*. Springer Berlin / Heidelberg. 3518, 256-261.

Gennaro, R., Rabin, M. O. and Rabin, T. (1998). Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. *Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing*, Puerto Vallarta, Mexico. 277716: ACM, 101-111.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. *41st annual ACM symposium on Theory of computing*. 169-178.

Goethals, B., Laur, S., Lipmaa, H. and Mielikainen, T. (2005). On private scalar product computation for privacy-preserving data mining. *7th International Conference on Information Security and Cryptology, Lecture Notes in Computer Science*. 3506, 104-120.

Goldreich, O. (1998). Secure multi party computation. *Working Draft Version,* 1.

Goldreich, O. (2004). *Foundations of Cryptography*. Cambridge Univ. Press.

Goldreich, O. (2005). Foundations of cryptography: a primer Found. *Trends Theor. Comput. Sci.,* 1, 1-116.

Goldreich, O. (2008). Encryption Schemes. *working draft.*

Goldreich, O., Micali, S. and Wigderson, A. (1987). How to play ANY mental game. *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, New York, New York, United States. ACM.

Goldreich, O., Micali, S. and Wigderson, A. (1991). Proofs that yield nothing but their validity or all languages in NP have zero knowledge proof systems. *JACM,* 38**,** pp 691–729.

Goldriech, O., Goldwasser, S. and Linial, N. (1991). Fault-Tolerant Computation in the Full Information Model. *32nd IEEE Symposium on Foundations of Computer Science*. 447–457.

Goldwasser, S. (1997). Multi party computations: past and present. *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, Santa Barbara, California, United States. 259405: ACM, 1-6.

Goldwasser, S. and Micali, S. (2006). Probabilistic encryption. *Comput. System Sci*.

Golle, P. and Juels, A. (2008). Dining cryptographers revisited. *Lecture Notes in Computer Science*.

Han, J., Pei, J., Yin, Y. and Mao, R. (2004). Mining frequent patterns without candidate generation. *Data Mining and Knowledge Discovery***,** 53-87.

Hipp, J., Güntzer, U. and Nakhaeizadeh, G. (2005). Algorithms for association rule mining. *SIGKDD Explorations***,** pp 1-58.

Hirt, M. and Maurer, U. (1997). Complete Characterization of Adversaries Tolerable in Secure Multi-party Computation. *16th Symposium on Principles of Distributed Computing*. ACM Press, 25–34.

Hirt, M. and Maurer, U. (2000). Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology***,** 13:31-60.

Hirt, M. and Sako, K. (2000). Efficient receipt-free voting based on homomorphic encryption. *In Advances in Cryptology - Proceedings of EUROCRYPT* volume 1807 of Lecture Notes in Computer Science.

Hong, J., Kim, J., Kim, J., Franklin, M. and Park, K. (2010). Fair threshold decryption with semi-trusted third parties. *International Journal of Applied Cryptography,* 2**,** 139 - 153.

Hong, J., Kim, J., Kim, J., Franklin, M. K. and Park, K. (2009). Fair Threshold Decryption with Semi-Trusted Third Parties. *Proceedings of the 14th Australasian Conference on Information Security and Privacy*, Brisbane, Australia. 1574955: Springer-Verlag, 309-326.

Horvitz, O. and Katz, J. (2007). Universally-composable two-party computation in two rounds. *Advances in Cryptology - (CRYPTO 2007)***,** 111-129.

Hussein, J. A. and AlMukhtar, M. A. (2009). Fair Exchange of Digital Signatures using RSA-based CEMBS and Offline STTP. *Journal of Computing,* 1**,** 87-91.

Inan, A., Kaya, S. V., SaygIn, Y., Savas, E., Hintoglu, A. A. and Levi, A. (2007). Privacy preserving clustering on horizontally partitioned data. *Data & Knowledge Engineering,* 63**,** 646-666.

Ioannidis, I. and Grama, A. (2003). An efficient protocol for yao's millionaires' problem. *36th Hawaii Internatinal Conference on System Sciences*. 6-9.

Ishai, Y., Prabhakaran, M. and Sahai, A. (2008). Founding Cryptography on Oblivious Transfer – Efficiently. WAGNER, D. (ed.) *Advances in Cryptology – CRYPTO 2008.* Springer Berlin / Heidelberg. 5157, 572-591.

Jain, A. and Hari, C. (2010). A New Efficient Protocol for k-out-of-n Oblivious Transfer. *Cryptologia,* 34**,** 282-290.

Jarecki, S. and Shmatikov, V. (2007). Efficient Two-Party Secure Computation on Committed Inputs. *EUROCRYPT*.

Jing, W., Huang, L., Yao, Y. and Xu, W. (2007). Privacy-preserving statistical quantitative rules mining. *Proceedings of the 2nd international conference on Scalable information systems*, Suzhou, China. 1366892: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 1-2.

Kabir, S. M. A., Youssef, A. M. and Elhakeem, A. K. (2007). On Data Distortion for Privacy Preserving Data Mining. *Canadian Conference on Electrical and Computer Engineering, (CCECE 2007).* Vancouver, BC

Kantarcioglu, M. and Clifton, C. (2002). Privacy preserving distributed mining of association rules on horizontally partitioned data. *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery***,** pp 24–31.

Keke, C. (2009). Privacy-Preserving Multiparty Collaborative Mining with Geometric Data Perturbation. *IEEE Transactions on Parallel and Distributed Systems,* 20**,** 1764-1776.

Laur, S. (2004). Special Course on Cryptology: Privacy-Preserving Frequent Itemset Mining on Horizontally Distributed Data. *Institute of Computer Science University of Tartu*.

Laur, S. and Lipmaa, H. (2006). On Security of Sublinear Oblivious Transfer. *Draft*.

Li, C.-T., Hwang, M.-S. and Lai, Y.-C. (2009). *Sixth International Conference on Information Technology: New Generations (ITNG '09)*, Las Vegas, NV 449 - 454

Li, Q., Chan, W. H. and Long, D.-Y. (2010). Semiquantum secret sharing using entangled states. *Physical Review A,* 82**,** 022303.

Lin, H.-Y. and Tzeng, W.-G. (2005). An Efficient Solution to the Millionaires' Problem Based on Homomorphic Encryption. IOANNIDIS, J., KEROMYTIS, A. and YUNG, M. (eds.). *Applied Cryptography and Network Security.* Springer Berlin / Heidelberg. 3531, 97-134.

Lindell, Y. and Pinkas, B. (2000). Privacy preserving data mining. *Advances in Cryptology—Proceedings of CRYPTO 2000 Lecture Notes in Computer Science vol 1880***,** pp 36–54.

Lindell, Y. and Pinkas, B. (2007). An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. *EUROCRYPT* 52-79.

Lu, Y. and Desmedt, Y. (2010). Improved Distinguishing Attack on Rabbit. *ISC***,** 17-23.

Luo, H., Zhao, Z. and Lu, Z. M. (2011). Joint secret sharing and data hiding for block truncation coding compressed image transmission. *Inform. Technol. J.,* 10**,** 681-685.

Ma, J. and Sivakumar, K. (2005). Privacy-Preserving Bayesian Network Parameter Learning. *4th WSEAS Int. Conf. on Computational Intelligence, Man-Machine Systems and Cybernetics*, Miami, Florida, USA. 46-51.

Ma, J. and Sivakumar, K. (2006). A Framework of Privacy-Preserving Bayesian Network Parameter Learning using Post Randomization. *WSEAS Transaction of Information Science and Applications,* 3.

Meng-chang, L. and Ning, Z. (2010). A solution to privacy-preserving two-party sign test on vertically partitioned data (P^2 $2NST_v$) using data disguising techniques. *2010 International Conference on Networking and Information Technology (ICNIT)* 11-12 June 2010. 526-534.

Michael, B.-O., Shafi, G. and Avi, W. (1988). Completeness theorems for non-cryptographic fault-tolerant distributed computation. *Proceedings of the twentieth annual ACM symposium on Theory of computing*, Chicago, Illinois, United States. ACM.

Mishra, D. K. and Chandwani, M. (2007). Extended Protocol for Secure Multiparty Computation using Ambiguous Identity. *WSEAS Transaction on Computer Research,* 2.

Mohassel, P. and Franklin, M. (2006). Efficiency Tradeoffs for Malicious Two-Party Computation. *Public Key Cryptography - PKC 2006.* 458-473.

Naor, M. and Pinkas, B. (1999). Oblivious Transfer and Polynomial Evaluation. *31st Symp. on Theory of Computer Science (STOC)*, Atlanta, GA. 245-254.

Naor, M. and Pinkas, B. (2001). Efficient Oblivious Transfer Protocols. *SODA 2001 (SIAM Symposium on Discrete Algorithms)*, Washington DC.

Naor, M. and Pinkas, B. (December 2000). Distributed Oblivious Transfer. *Advances in Cryptology -- Asiacrypt '00, LNCS 1976, Springer-Verlag,* 200-219.

Nielsen, J. and Orlandi, C. (2009). LEGO for Two-Party Secure Computation. REINGOLD, O. (ed.) *Theory of Cryptography.* Springer Berlin / Heidelberg. 5444, 368-386.

Otsuka, A. and Imai, H. (2010). Unconditionally Secure Electronic Voting. CHAUM, D., JAKOBSSON, M., RIVEST, R., RYAN, P., BENALOH, J., KUTYLOWSKI, M. and ADIDA, B. (eds.). *Towards Trustworthy Elections.* Springer Berlin / Heidelberg. 6000, 107-123.

Paillier, P. (1999). Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. *EUROCRYPT,* 223-238.

Park, H.-A., Lee, D., Lim, J. and Cho, S. (2007). PPIDS: Privacy Preserving Intrusion Detection System. YANG, C., ZENG, D., CHAU, M., CHANG, K., YANG, Q., CHENG, X., WANG, J., WANG, F.-Y. and CHEN, H. (eds.). *Intelligence and Security Informatics.* Springer Berlin / Heidelberg. 4430, 269-274.

Peng, B., Geng, X. and Zhang, J. (2010). Combined data distortion strategies for privacy-preserving data mining. *3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, Chengdu IEEE, 1, 572-576.

Peng, K., Aditya, R., Boyd, C., Dawson, E. and Lee, B. (2005). Multiplicative Homomorphic E-Voting. CANTEAUT, A. and VISWANATHAN, K. (eds.). *Progress in Cryptology - INDOCRYPT 2004.* Springer Berlin / Heidelberg. 3348, 1403-1418.

Peter, B., Dan Lund, C., Ivan, D., rd, Martin, G., Thomas, J., Mikkel, K., igaard, Janus Dam, N., Jesper Buus, N., Kurt, N., Jakob, P., Michael, S. and Tomas,

T. (2009). Secure Multiparty Computation Goes Live. *Financial Cryptography and Data Security: 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers.* Springer-Verlag. 325-343.

Pinkas, B., Schneider, T., Smart, N. and Williams, S. (2009). Secure Two-Party Computation Is Practical. *Advances in Cryptology – ASIACRYPT 2009.* 250-267.

Pinto, C. B., Dowsley, R., Morozov, K. and Nascimento, C. A. (2009). Achieving Oblivious Transfer Capacity of Generalized Erasure Channels in the Malicious Model. *Cryptology ePrint Archive*.

Rabin, M. O. (1981). How to exchange secrets by oblivious transfer. *Technical Report TR-81.* Aiken Computation Laboratory, Harvard University.

Rebollo-Monedero, D. and Forne, J. (2010). Optimized Query Forgery for Private Information Retrieval. *IEEE Transactions on Information Theory,* 56**,** 4631 - 4642.

Rivest, R., Shamir, A. and Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM,* 21 120-126.

Rizvi, S. J. and Haritsa, J. R. (2002). Maintaining data privacy in association rule mining. *Proceedings of 28th International Conference on Very Large Data Bases***,** pp 682–693.

Rozenberg, B. and Gudes, E. (2003). privacy preserving frequent item set mining in vertically partitioned databases. *DBSec***,** pp 91–104.

Saygin, Y., Verykios, V. S. and Elmagarmid, A. K. (2002). Privacy preserving association rule mining. *Research Issues in Data Engineering (RIDE)***,** pp 151–158.

Schaffner, C. (2010). Simple protocols for oblivious transfer and secure identification in the noisy-quantum-storage model. *Physical Review A,* 82**,** 032308.

Shamir, A. (1979). How to share a secret. *Communications of the ACM***,** 22(11) 612–613.

Shao, Z. (2010). Fair exchange protocol of Schnorr signatures with semi-trusted adjudicator. *Computers & Electrical Engineering*, 36**,** 1035-1045.

Shen, Y., Han, J. and Shan, H. (2010). The Research of Privacy-Preserving Clustering Algorithm. *Third International Symposium on Intelligent Information Technology and Security Informatics (IITSI)*, Jinggangshan 324 - 327

Sherman, C., Jie Han, L. and Lakshminarayanan, S. (2009). Two-Party Computation Model for Privacy Preserving Queries over Distributed Databases. *Networks and Distributed Systems Security (NDSS)*.

Stanley, R., Oliveira, M. and Osmar, R. (2002). Privacy preserving frequent itemset mining. *Proceedings of the IEEE international conference on Privacy, security and data mining - Volume 14*, Maebashi City, Japan. Australian Computer Society, Inc.

Tzeng, W.-G. (2002). Efficient 1-Out-n Oblivious Transfer Schemes. NACCACHE, D. and PAILLIER, P. (eds.). *Public Key Cryptography***.** Springer Berlin / Heidelberg. 2274, 359-362.

Urabe, S., Wang, J., Kodama, E. and Takata, T. (2007). A High Collusion-Resistant Approach to Distributed Privacy-preserving Data Mining. *IPSJ Digital Courier,* 3**,** 442-455.

Vaidya, J. and Clifton, C. (2004). Privacy preserving naive Bayes classifier on vertically partitioned data. *In Proceedings of 2004 SIAM International Conference on Data Mining*.

Vanishree, H. and Iyengar, S. R. S. (2009). A novel efficient m-out-of-n oblivious transfer scheme. *IET Seminar Digests,* 2009**,** 64-64.

Wang, M.-N., Yen, S.-M., Wu, C.-D. and Lin, C.-T. (2006). Cryptanalysis on an Elgamal-like cryptosystem for encrypting large messages. *Proceedings of the 6th WSEAS International Conference on Applied Informatics and Communications*, Elounda, Greece.

Wong, K. S. and Kim, M. H. (2010). Semi-trusted collaborative framework for multi-party computation. *KSII Transactions on Internet and Information Systems,* 4.

Woodruff, D. P. (2007). Revisiting the Efficiency of Malicious Two-Party Computation. *Proceedings of the 26th annual international conference on Advances in Cryptology*, Barcelona, Spain. Springer-Verlag.

Wright, R. (2008). Progress on the PORTIA Project in Privacy Preserving Data Mining. *data surveillance and privacy protection workshop*.

Wright, R. N. and Yang, Z. (2004). Privacy-preserving Bayesian network structure computation on distributed heterogeneous data. *Proceedings of KDD 2004***,** pp 713–718.

Yang, Z. (2007). *Distributed Protocols for Data Privacy***.** Doctor of Philosophy, Stevens Institute of Technology.

Yao, A. C.-C. (1982). Protocols for Secure Computations. *FOCS***,** 160-164.

Yao, A. C.-C. (1986). How to generate and exchange secrets. *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society.

Yehuda, L., Benny, P. and Nigel, P. S. (2008). Implementing Two-Party Computation Efficiently with Security Against Malicious Adversaries. *Proceedings of the 6th international conference on Security and Cryptography for Networks*, Amalfi, Italy. Springer-Verlag.

Yu, J. X., Chong, Z., Lu, H., Zhang, Z. and Zhou, A. (2006). A false negative approach to mining frequent itemsets from high speed transactional data streams. *Inform. Sci*, 176**,** 1986-2015.

Zaki, M. J. (2006). Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering***,** 372-390.

Zhong and Sheng (2007). Privacy-preserving algorithms for distributed mining of frequent itemsets. *Information Sciences,* 177**,** 490-503.