

# Privacy-Preserving Statistical Analysis Methods and their Applications on Health Research

by

© *Ahoora Sadeghi Boroujerdi*

A thesis submitted to the  
School of Graduate Studies  
in partial fulfilment of the  
requirements for the degree of  
Master of *Computer Science*

Department of *Computer Science*  
Memorial University of Newfoundland

*October 2016*

St. John's

Newfoundland

## Abstract

Privacy consideration in health data usually prevents researchers and other data users from conducting their research. Also, data is distributed through various health organizations such as hospitals, thus gathering distributed health information becomes impractical. Various approaches have been proposed to preserve the patients privacy, whilst allowing researchers to perform mathematical operations and statistical analysis methods on health data, such as anonymization and secure computation. Data anonymization reduces the accuracy of the original data; hence the final result would not be precise enough. In addition, there are several known attacks on anonymized data, such as using public information and background knowledge to re-identify the original data. On the other hand, secure computation is more precise and the risk of data re-identification is zero; however, it is computationally less efficient than data anonymization. In this thesis, we implemented a web-based secure computation framework and propose new secure statistical analysis methods. Using the proposed web application, researchers and other data users would be able to perform popular statistical analysis methods on distributed data. They will be able to perform mathematical operations and statistical analysis methods as queries through different data owners, and receive the final result without revealing any sensitive information. Digital Epidemiology Chronic Disease Tool (DEPICT) database, which contains real patients information, will be used to demonstrate the applicability of the web application.

**Keywords** Secure Multiparty Computation; Web-Based Framework, Privacy-Preserving; Homomorphic Encryption, Secure Statistical Analysis Methods

## Acknowledgements

First, I would like to thank my mentor and supervisor Dr. Saeed Samet. Prof. Samet was always there for me and was always helpful and understanding when I ran into a problem or have a question about my research. I would have not been able to do this research if it was not for him. He allowed me to work on the research on my own, but guided me on the right direction whenever I needed it.

I would also like to thank the experts who were involved and contributed in this research: Dr. Shabnam Ashghari, Dr. Gerard Farrell and Mr. Oliver Hurley for their thoughtful comments, suggestions and contributions.

I thank all my friends for their continuous support through all the challenges that I faced throughout my academic life at Memorial University of Newfoundland.

I consider myself the luckiest person for having the best wife anyone can think of. I can not think of any words to express my very profound gratitude towards my wife, Nadia. Her strength, love and understanding gave me everything that I needed throughout my years of study. This accomplishment would not have been possible without her. Thank you.

Last but not least, I would like to thank my family: my parents and my brothers for encouraging me to continue my study towards Masters and for supporting me throughout writing this thesis and my life in general.

# Contents

<b>Abstract</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>List of Tables</b>	<b>vii</b>
<b>List of Figures</b>	<b>viii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Preliminaries and Definitions</b>	<b>5</b>
2.1 Statistical Analysis Methods . . . . .	5
2.2 Statistical Analysis Methods on Health Research . . . . .	10
2.3 Homomorphic Encryption . . . . .	11
2.4 Paillier Cryptosystem . . . . .	13
<b>3 Related Work</b>	<b>16</b>
<b>4 Proposed Approach</b>	<b>25</b>
4.1 Data Distribution . . . . .	25

4.2	Secure Building Blocks . . . . .	29
4.3	Statistical Analysis Methods . . . . .	32
4.3.1	Secure Student $t$ -test . . . . .	34
4.3.2	Secure Linear Regression . . . . .	35
4.3.3	Secure Logistic Regression . . . . .	38
4.4	Complexity Assumptions . . . . .	43
4.4.1	Building Blocks Complexity . . . . .	44
4.4.1.1	Secure Multiplication . . . . .	44
4.4.1.2	Secure Addition . . . . .	45
4.4.1.3	Secure Dot Product . . . . .	46
4.4.2	Secure Statistical Analysis Methods Complexity . . . . .	47
4.4.2.1	Secure Mean . . . . .	47
4.4.2.2	Secure Variance . . . . .	48
4.4.2.3	Secure Skewness . . . . .	48
4.4.2.4	Secure Correlation . . . . .	49
4.4.2.5	Secure Student's $t$ -test . . . . .	50
4.4.2.6	Secure Linear Regression . . . . .	51
4.4.2.7	Secure Logistic Regression . . . . .	51
4.5	Statistical Analysis Methods in Healthcare . . . . .	52
<b>5</b>	<b>Implementation</b>	<b>54</b>
5.1	Java Classes . . . . .	58
5.1.1	Database Helper Class (DBHelper.java) . . . . .	58
5.1.2	Global Variables Class (Globals.java) . . . . .	61

5.1.3	System Messages Class (Message.java) . . . . .	61
5.1.4	Utility Class (Utility.java) . . . . .	61
5.1.5	Convert Class (Convert.java) . . . . .	62
5.1.6	User Class (User.java) . . . . .	63
5.1.7	Paillier Class (Paillier.java) . . . . .	68
5.1.8	Query Analyzer Class (QueryAnalyzer.java) . . . . .	69
5.1.9	Secure Computation Class (SecureComputation.java) . . . . .	70
5.1.10	Data Exchange . . . . .	73
5.2	Controllers . . . . .	74
5.2.1	Main Controller (index.jsp) . . . . .	74
5.2.2	Menu Controller (leftColumn.jsp) . . . . .	74
5.2.3	Login Controller (loginCheck.jsp) . . . . .	75
5.2.4	Logout Controller (logoutC.jsp) . . . . .	75
5.2.5	Secure Computation Controller (queryC.jsp) . . . . .	75
5.3	Web Service . . . . .	76
5.4	Views . . . . .	77
<b>6</b>	<b>Results</b>	<b>85</b>
6.1	Performance Analysis . . . . .	88
<b>7</b>	<b>Conclusion and Future Work</b>	<b>93</b>
	<b>Bibliography</b>	<b>95</b>
<b>A</b>	<b>UML Diagrams</b>	<b>104</b>

# List of Tables

1	2x2 Contingency Table . . . . .	8
2	Performance Results for the Pre-processing Stage . . . . .	86
3	Performance Results for the Cryptosystem and Secure Addition . . .	86
4	Performance Results for the Protocols Including Fetching Data from Database . . . . .	87
5	Performance Results for the Protocols when Data is Stored in Memory	87
6	Secure Protocols Performances with no conditions and 10000 records	89
7	Secure Protocols Performances with no conditions and 100000 records	89
8	Secure Protocols Performances with conditions and 10000 records . .	90
9	Secure Protocols Performances with conditions and 100000 records . .	91

# List of Figures

1	k-anonymity example . . . . .	19
2	Clifton secure sum protocol . . . . .	23
3	subset data distribution . . . . .	26
4	partial data distribution . . . . .	27
5	Orocess Flow of the Protocol . . . . .	33
6	Use Case Diagram . . . . .	55
7	Sequence Diagram . . . . .	56
8	Desktop Application . . . . .	57
9	Mean Query Page . . . . .	79
10	Mean Query Result . . . . .	80
11	Correlation Query Page . . . . .	81
12	Correlation Query Result . . . . .	82
13	Linear Regression Query Page . . . . .	83
14	Linear Regression Query Result . . . . .	84
15	registration use case diagram . . . . .	105
16	login use case diagram . . . . .	106

17	key generation use case diagram . . . . .	107
18	query submission use case diagram . . . . .	108
19	secure computation database scheme . . . . .	109
20	secure computation class diagram . . . . .	110

# Chapter 1

## Introduction

Preserving the security and privacy of health information plays an important role in the modern era. Technology grows every day and the need to protect confidential data has become essential. At the same time, researchers and scientists should be able to access these data for research purposes, to execute mathematical operations or statistical analysis methods on data. For many years, researchers had to physically go to health data centers in order to conduct their research. This is still a common practice; however, two settings introduced for privacy-preserving have brought more options to make this process easier. The first approach is anonymization and randomization such as generalization, suppression, and random sampling, and the second one is secure (multi-party) computation. There are various approaches for data anonymization such as optimal k-anonymization [7] and fast data anonymization [27]. When anonymization is used, no confidential data is revealed and statistical methods can be executed. The results taken from de-identified data are not precise enough when privacy is in high concern, thus using another approach seems essential.

Secure computation leads to more precise and secure results; therefore, researchers would be able to perform their own statistical methods on the data without revealing any confidential information. The risk of re-identification is close to zero, but it could be computationally prohibitive comparing to performing statistical analysis methods on anonymized data.

Andrew Yao [64] introduced the concept of secure computation in 1982. The idea was to perform a function on two parties' private data, and only return the result to each party (or a third party) without revealing any other information from one to the other. Not long after, Yao's approach was generalized by Goldreich, Micali and Wigderson for multiple parties [29]. This means that instead of only two parties, more than two parties ( $n$  where  $n \geq 3$ ) parties can participate.

Secure computation guarantees that no sensitive information would be revealed, but allows the end-users to perform a function on their private data. Secure computation can be done using various approaches, and several implementations of secure computations have been proposed and developed in recent years [37, 5]. Also, various privacy-preserving protocols with diverse functionalities have been built based on the above approaches. In addition, cryptography techniques, such as homomorphic encryption, could be used for secure computation. There are two types of homomorphic encryption systems; partially homomorphic cryptosystems such as Paillier [46], Elgamal [22], and RSA [48], and fully homomorphic encryption systems such as the work proposed by Gentry [26].

Ideally, any mathematical operation can be performed on encrypted data using fully homomorphic cryptosystems. Gentry [26] proposed the most popular fully homomorphic encryption system in 2009. The performance of the proposed cryptosystem

does not live up to other cryptosystems, which makes it impractical for long messages. However, not all mathematical operations can be performed using partially homomorphic encryption systems. Gentry [26] scheme supports both addition and multiplication operations and is based on somewhat homomorphic encryption scheme. This is still impractical when the security level increases. Cramer et al. [14] scheme is based on homomorphic threshold cryptosystem, which can be used for multi-party computation, but there is an issue with their approach. Long messages cannot be split, because the chosen ciphertext property is not preserved in this way. Domingo Ferrer [18] proposed another homomorphic encryption scheme, but the ciphertext message space is too large and is impractical for long messages. Although fully homomorphic encryption systems are making huge improvements for secure computation, no practical approach has been proposed yet. When dealing with sensitive data such as health information, there are many factors to consider other than the security of the system such as the usage and disclosure of personal health information.

Personal health information is a sensitive type of personal information that can be shared for multiple purposes such as research, treatment and care. The collection, use and disclosure of personal health information has been addressed by different privacy acts such as Personal Health Information Act (PHIA) and Personal Health Information Protection Act (PHIPA). These acts consists of sets of rules in which all data custodians, individuals, agents, and health organizations should follow. Every person who delivers healthcare services or product should follow defined rules. There exists seven categories in the definition. Under this act, each individual or organization that provides technology to a data custodian, must follow the restrictions on how to use and disclose personal health information. Also, all individuals or organizations that

are the recipient of personal health information, must follow the act. The act protects personal health information in oral or written form that are related to a person's physical or mental health, identification, payment related to healthcare and other information that are related to an individual's information. These rules restrict and prevent health organizations to share their information for research purposes. Also, health organizations usually resist on sharing information for security and beneficial reasons, but they are willing to provide secure services in order to get an aggregated result.

A practical and secure solution has been introduced by Samet et al. [51], in which data owners could securely and jointly perform a privacy-preserving protocol by only exchanging encrypted values between themselves. At the end, each of them will send their portion of the final results to the researcher to combine the received shares and construct the final results of their query. In this thesis, we focus on implementing a web-based framework based on various works done by Samet such as [52, 53, 50, 51], which can be improved by adding more statistical analysis methods modified for multi-party computation. Various features have been added to this framework, such as including an authorized server for user access control or system parameterization. The outline of this thesis is as follow: Chapter 2 covers basic definitions and preliminaries. In Chapter 3, we review background and related work. The proposed approach, research methods and various aspects of the system such as configurations are discussed in Chapter 4. In Chapter 5, we thoroughly explain the implementation of the web-based framework, system parameters, and features. Experimental results will be covered and shown in Chapter 6, followed by conclusions and future work in Chapter 7.

# Chapter 2

## Preliminaries and Definitions

In this chapter, we will describe the terminologies, techniques, and concepts used in our system to make sure that reader will have basic understanding of our system design.

### 2.1 Statistical Analysis Methods

Statistical analysis is the science of collecting, interpreting and presenting large amount of data to find out about the existing patterns and trends. Statistical analysis is fundamental to all researches that use statistics as a methodology. In social sciences, engineering, healthcare and many other fields, statistics are applied every day. There exists various statistical analysis methods such as arithmetic mean, variance, standard deviation, student's  $t$ -test, linear regression and logistic regression. Therefore, each can be used to interpret data in a unique way. Some of statistical analysis methods are as below:

1. **Mean:** The arithmetic mean is an average score which is the sum of individual scores divided by the number of those individuals. If our sample data consists of values  $x_1, x_2, \dots, x_n$ , mean (denoted by  $\bar{x}$ ) is as below:

$$\bar{x} = \frac{(x_1 + x_2 + \dots + x_n)}{n} \quad (2.1)$$

2. **Standard Deviation** [9]: Standard deviation is to measure the dispersion or deviation of a set of data from its mean and is a measure of the spread of scores. To calculate standard deviation, variance has to be calculated first and the square root of variance is standard deviation. Variance is also a method to measure the spread between numbers in a data set. To calculate the variance, the differences between each number in the data set and the mean is measured and squared. Then the summation of these values is calculated and divided by the number of the values in the set. If our sample data consists of values  $x_1, x_2, \dots, x_n$ , variance (denoted by  $\sigma^2$ ) is as below:

$$\sigma^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n} \quad (2.2)$$

Thus, standard deviation (denoted by  $\sigma$ ) is:

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}} \quad (2.3)$$

3. **Skewness** [41]: When the normal distribution of a set of data is measured, the distribution might have fewer data on the right or the left, which is said that the distribution is skewed to the right or left respectively. Skewness is a method to

calculate the asymmetry of a set of data in normal distribution. If our sample data consists of values  $x_1, x_2, \dots, x_n$ , skewness (denoted by  $\gamma$ ) is as below:

$$\gamma = \frac{\sum_{i=1}^n (x_i - \bar{x})^3}{n\sigma^3} \quad (2.4)$$

4. **Correlation** [24]: Correlation is a statistical method to measure how strong two sets of variables are linked. Also, it can be used to measure the movement (increase or decrease) of the second variable when the first variable moves. Correlation ranges between  $-1$  and  $+1$ . When correlation is positive, it can be implied that the sets of variables move in the same direction (increase or decrease together). Also, when correlation is negative, it shows that the variables move in the opposite direction (when one increases, the other decreases). If our sample data consists of values  $x_1, x_2, \dots, x_n$  for the first variable and  $y_1, y_2, \dots, y_n$  for the second variable, correlation (denoted by  $\rho$ ) is as below:

$$\rho = \frac{(\sum_{i=1}^n x_i * y_i) - n * \bar{x} * \bar{y}}{n * \sigma_x * \sigma_y} \quad (2.5)$$

5. **Chi-Square Test** [40]: Chi-Square is a test to compare the observed data with the expected result according to a hypothesis. Also, it can be used to view the distribution differences between categorical variables. Thus, the variables' types should be categorical such as "yes" and "no" or "0" and "1". For simplicity of the mentioned method, we imagine a 2x2 contingency table for two categorical variables and two data types. The contingency table is shown in Table 1.

For the 2x2 contingency table shown in Table 1, Chi-Square (denoted by  $\chi^2$ )

Variables	Data Type 1	Data Type 2
Category 1	$x_1$	$x_2$
Category 2	$x_3$	$x_4$

Table 1: 2x2 Contingency Table

is as below:

$$\chi^2 = \frac{(x_1 * x_4 - x_2 * x_3)^2 * (x_1 * x_2 * x_3 * x_4)}{(x_1 + x_2) * (x_3 + x_4) * (x_2 + x_4) * (x_1 + x_3)} \quad (2.6)$$

When  $\chi^2$  is close to zero, we denote that there is no significant difference between the observed result and the expected result.

6. **Student's t-test** [56]: A  $t$ -test is an examination of two samples' means. In students  $t$ -test, both samples should follow the normal distribution if the null hypothesis is true and examines the difference between two samples. It is mostly used when there are two variables and one of them is a measurement variable and the other is a categorical variable. For a sample of size  $n$  with mean value of  $\bar{x}_1$  and standard deviation  $\sigma$  drawn from a random population with mean value of  $\bar{x}_2$  and unknown standard deviation, Student's  $t$ -test (denoted by  $t$ ) is as below:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sigma/\sqrt{n}} \quad (2.7)$$

In student's  $t$ -test, the degree of approximation depends on parent's population closeness to normal distribution and also on the sample size.

7. **Linear Regression** [45]: Linear regression is a predictive analysis to model the relationship between two variables where one variable is a dependent variable

and the other variable is independent. Linear regression finds a linear function that fits a given set of data points in the dataset that best describes the bivariate data. If  $X$  is the independent variable and  $Y$  is the dependant variable, a linear regression line equation is as below:

$$Y = a + bX \quad (2.8)$$

where  $a$  is a constant value when  $x = 0$  and  $b$  is the regression coefficient. When  $x$  is the value of an independent variable, the predicted value (denoted by  $\hat{y}$ ) is as below:

$$\hat{y} = a + bx \quad (2.9)$$

$a$  and  $b$  are calculated as below:

$$b = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad (2.10)$$

$$a = \bar{y} - b * \bar{x} \quad (2.11)$$

8. **Logistic Regression** [33]: Logistic regression is a statistical method to model the relationship between one dependant variable which is dichotomous and one or more independent variable(s). Logistic regression can be described as a special case for generalized linear model which is based on different assumptions than linear regression. Unlike linear regression that is used for both continuous or categorical variables, logistic regression is used for binary dependent variables

## 2.2 Statistical Analysis Methods on Health Research

Statistical methods have plenty of utilizations such as healthcare resources and cost analysis [43], quality improvement [23], health assessment [6] and many other applications. As an example, we can simplify the outpatient clinic problem discussed in [23]. The management of an outpatient clinic was notified that the average time between the patient arrival and the blood test or electrocardiogram (ECG) of the patient is around 30 minutes. The management decided to monitor the average waiting time of patients in each day of the week for several weeks. Afterwards, the mean and standard deviation of the waiting time was calculated for each day of the week. By comparing the calculated values for each day, they found out that the average waiting time of the patients on Friday is a lot more than the other days. Thus, they monitor the patients on Friday and found out that there are more ECG patients on Fridays than blood test patients comparing other days of the week. ECG patients require more examination time than blood test patients. Thus, the average waiting time on Fridays was significantly more than the other days. Based on the given analysis, the management decided to dedicate more doctors and nurses on Fridays for ECG patients and as a result the average waiting time of patients reduced to 15 minutes. More real life examples can be found at Section 4.5.

## 2.3 Homomorphic Encryption

In algebra, homomorphism is a map between two groups respecting the structure of each group. A group is a set that includes a law and an operation ( $\Delta$ ). Assuming we have two groups such as  $(G, \Delta)$  and  $(H, \circ)$  where  $\Delta$  and  $\circ$  represent operations on set  $G$  and  $H$  respectively, a group homomorphism from  $G$  to  $H$  is defined as below:

$$f(a\Delta b) = f(a) \circ f(b),$$

where  $f$  is a function from  $G$  to  $H$  for all  $a$  and  $b$  in  $G$ . Homomorphic encryption is a type of encryption that allows specific types of computations on the plaintext to be carried on the ciphertext. If we assume that the plaintexts and ciphertexts create two homomorphism groups such as  $(P, \Delta)$  and  $(C, \circ)$  respectively, we have:

$$E(a)\Delta E(b) = E(a \circ b), \text{ for all } a \text{ and } b \text{ in } P \quad (2.12)$$

It also allows computations to be performed on encrypted data without revealing confidential information about the original data. However, if the private key gets compromised, data can be leaked by decrypting the encrypted values. There are two types of Homomorphic Encryption systems: Fully Homomorphic and Partially Homomorphic cryptosystems. A cryptosystem is called a fully homomorphic encryption system if it supports arbitrary number of operations such as multiplication and addition to implement any functions that are needed. Gentry cryptosystem [26] is the first practical fully homomorphic cryptosystem which supports both multiplication and addition. Partially homomorphic cryptosystems only support one operation such as addition or multiplication. There are several partially homomorphic cryptosystems such as El-Gamal cryptosystem [22] which supports multiplication or Paillier

cryptosystem [46] which supports addition. In addition, there exist somewhat homomorphic encryption systems. These cryptosystems support a limited number of operations. Fully homomorphic cryptosystems such as Gentry cryptosystem [26] have tremendous overhead and other approaches are not practical. Thus, somewhat fully homomorphic encryption systems such as Brakerski's scheme [11] which is based on the ring learning with errors assumption could be used for most of the problems that do not need an arbitrary number of computations. Many works have been done in somewhat fully homomorphic encryption field for secure multi-party computation such as Damgard et al. [15] multi-party computation from somewhat homomorphic encryption and Boneh et al. [10] private database queries using somewhat homomorphic encryption. Most of the mentioned encryption systems are not practical or secure enough for multi-party computations. Also, the encrypted data is a pair of numbers which makes fully homomorphic encryption systems more complicated and impractical to use. Paillier cryptosystem is secure against chosen plaintext attack. Also, Paillier encryption uses randomness in its encryption method which produces different ciphertexts for the same plaintext. Because Paillier is an additive homomorphic encryption system, we are unable to implement most of the statistical analysis methods. Thus, we use method proposed in [53] for secure multiplication and the method of [28] for secure dot product to empower Paillier cryptosystem with various mathematical operations.

## 2.4 Paillier Cryptosystem

Paillier cryptosystem named after Pascal Paillier is a modular public key scheme which is also an additive homomorphic cryptosystem. Like any other public key scheme, it has its own key generation phase, encryption phase using public keys and decryption phase using private keys. Key generation is as follow:

1. To construct public keys, two randomly generated large prime numbers  $p, q$  are needed. For simplicity, these two numbers can have equal length.
2. Then we calculate their product such that:

$$n = p * q \quad (2.13)$$

3. We also calculate the least common multiple of  $p - 1$  and  $q - 1$  such that:

$$\lambda = lcm(p - 1, q - 1) \quad (2.14)$$

4. A random integer  $g$  should be selected from  $Z_{n^2}$ . To ensure that  $g$  is divisible by  $n$ , the relationship described by the below equation should hold between the chosen and used variables:

$$\mu = (L(g^\lambda \bmod n^2))^{-1}(\bmod n) \quad (2.15)$$

where  $L$  is a function such that:

$$L(u) = \frac{u - 1}{n} \quad (2.16)$$

5. The public keys are  $n$  and  $g$ .
6. The private keys are  $\mu$  and  $\lambda$ .

After generating public keys and private keys, we will be able to encrypt and decrypt a message and ciphertext respectively. The encryption process is as follow:

1. We choose message  $m$  from  $Z_n$ .
2. Then we select a random number  $r$  from  $Z_n^*$  ( $Z_n^*$  being the invertible elements of  $Z_n$ ).
3. Encrypting a message is as follow:

$$c = E(m) = g^m * r^n \pmod{n^2} \quad (2.17)$$

The decryption process would be as follow:

1. We assume that ciphertext  $c$  is from  $Z_{n^2}^*$  and previously created by the cryptosystem (it ensures us that  $c$  is invertible in  $Z_{n^2}^*$ ).
2. Decrypting the ciphertext is as follow:

$$m = D(c) = L(c^\lambda \pmod{n^2}) * \mu \pmod{n^2} \quad (2.18)$$

Here is an example of Paillier Cryptosystem. For ease of calculations, small prime numbers will be chosen for this purpose. Suppose we choose  $p = 13$  and  $q = 11$  for our prime numbers. Thus,  $n = 13 * 11 = 143$ . Furthermore, we have to choose a random  $g$  and calculate  $\lambda$  and  $\mu$  as below:

$$g = 67, \lambda = lcm(12, 10) = 60, \mu = (L(67^{60} \pmod{20449}))^{-1} \pmod{143} = 124 \quad (2.19)$$

For example, we want to encrypt 7 as our message. First we have to select a random number and then encrypt the message as below:

$$r = 5 \Rightarrow c = E(7) = 67^7 * 5^{143} \pmod{20449} = 14271 \quad (2.20)$$

Now we want to decrypt a ciphertext such as 14271. The process is as below:

$$m = D(14271) = L(14271^{60} \bmod 20449) * 124 \bmod 143 = 7 \quad (2.21)$$

# Chapter 3

## Related Work

Many privacy-preserving techniques tend to transform information in order to preserve privacy. These methods reduce the accuracy of the data and cause information loss. Randomization and anonymization are some examples of transformation methods. Randomization is done by adding noise components to the original data in a way that the variance of the added noise is large enough. By adding noise, original data cannot be derived from distorted data. Randomization has been used for data distortion and found its way to privacy preserving in data mining [5]. Noise can be added to the values in database such as the method proposed by Traub and Yemini [58] in a way that each vector in database is replaced by another vector which creates a random perturbed vector. Also, noise can be added to the final result of the query such as the method proposed by Beck [8]. Although randomization is relatively simple and no knowledge is needed for the distribution of data, it is more vulnerable to adversarial attacks [2]. More noise should be added to the original data in order to provide more security using randomization. As a result, it reduces the

accuracy of the data for data mining purposes. In the past, many data owners created statistical databases using randomization techniques for research purposes [63]. However, creating statistical databases became time consuming and costly due to tremendous increase in demand for person-specific data for various applications such as data mining and fraud detection. Moreover, adding noise to these databases for preserving the privacy led to inaccurate results.

Data anonymization is the process of de-identifying sensitive data in order to protect data without any change to the data type. One of the popular techniques for anonymization is inference in multi-level databases [44]. In multi-level database, the data is restricted in a way that only lower classified information will be released in a way that higher classified information could not be revealed. In a more advance approach which was proposed by Denning and Lunt [17], multiple security classifications could be defined for multiple types of users and multilevel databases could be created for each security classification.


In 1999, Agrawal [4] mentioned that an important field of future research could be the development of techniques to overcome privacy concern. In 2000, Agrawal and Srikant [5] proposed a method for privacy preserving in the data mining field. The idea was to let users provide a value for sensitive fields using Value-Class Membership method or Value Distortion method and reconstructing the original distribution. In their method, the values for an attribute are classified into different groups in a way that they are mutually exclusive. So, by using discretization some intervals are created where each value can be stored in a specific interval. In this way, true value does not have to be stored in the database. For example, age may be discretized into five different intervals such as less than 18, between 18 and 30, between 30 and 50

and more than 50. By using this discretization, true values for each attribute will be hidden. Also, values can be distorted instead of being classified. A truly random number will be added to each value in order to prevent snoopers from estimating the true value through multiple queries. The random value can be generated using Uniform Distribution or Gaussian Distribution. It has been shown that discretization is less efficient than distribution [5]. In order to have the same level of privacy using discretization comparing to uniform distribution, interval width should be increased significantly which results in inaccurate model. Also, it has been shown that the Gaussian distribution provides more privacy at higher confidence level compared to uniform distribution [5]. Thus, the Gaussian distribution has been used in their approach. In this system, the original distribution should be reconstructed in order to be useful. The original data distribution can be reconstructed using decision tree classification over randomized data as shown in [5].

In modern era, data is distributed through various data owners and also there is redundancy between many data owners, thus data owners have incomplete knowledge over one another. As a result, releasing information without proper knowledge of other released data may cause revealing sensitive information. Also, suppression is used for designing multilevel databases which can reduce the quality of the final result.

Another technique for data anonymization is k-anonymity, which was introduced by Sweeney [57] in 2002. K-anonymity would guarantee that each record of a data table is similar to at least k-1 other records on the data table (in terms of Quasi-identifiers). An example of k-anonymity is shown in Figure 1. There are two common methods for k-anonymity; suppression and generalization. In suppression, the values of a spe-

cific attribute will be removed completely. In generalization, the values of a specific attribute are generalized to a range. K-anonymity reduces the risk of information



Id	Postal Code	Age	Nationality	Condition
1	A1A 3K8	22	Canadian	Heart Disease
2	A1A 2B7	27	Iranian	Cancer
3	A1A 6K9	23	American	Cancer
4	A1B 2C4	45	Japanese	Alzheimer's
5	A1B 4B5	64	Chinese	Heart Disease
6	A1B 1C8	42	Iranian	Alzheimer's
7	A1B 4K5	40	Canadian	Cancer
8	A1C 2S6	33	Russian	Heart Disease
9	A1C 5B5	38	Chonese	Cancer

Id	Postal Code	Age	Nationality	Condition
1	A1A ***	<30	*	Heart Disease
2	A1A ***	<30	*	Cancer
3	A1A ***	<30	*	Cancer
4	A1B ***	≥40	*	Alzheimer's
5	A1B ***	≥40	*	Heart Disease
6	A1B ***	≥40	*	Alzheimer's
7	A1B ***	≥40	*	Cancer
8	A1C ***	3*	*	Heart Disease
9	A1C ***	3*	*	Cancer

Figure 1: k-anonymity example

identification, but it also reduces the accuracy of the results on the applications based on the mentioned methods. Although it has been shown that k-anonymity is NP-hard [42], there are some heuristic methods to solve the problem such as Bayardo and Agrawal k-optimize algorithm [7]. K-anonymity does not include any randomness, so attackers are able to extract individual and sensitive information by using inference attacks. K-anonymity is not secure against various attacks such as Homogeneity Attack and Background Knowledge Attack. Also, k-anonymity should not be used for high dimensional datasets [1]. There are some partial solutions such as I-diversity. When I-diversity is used, the granularity of a data set is reduced. I-diversity maintains the diversity of sensitive fields especially when the sensitive values within a group exhibit homogeneity. It has been shown that I-diversity is secure against background knowledge attacks and homogeneity attacks [38]. Attribute values might be skewed in a dataset, thus there might not be a feasible representation for I-diversity. T-closeness [36] is another solution to overcome issues with k-anonymity. T-closeness

is a refinement on I-diversity by treating each value of an attribute separately and is based on the distribution of values for each attribute. Both I-diversity and T-closeness reduce the utilization of the data.

There are many ways to anonymize data, so data might not be anonymized in a similar way between various data owners (i.e. hospitals). In many cases, data is distributed through different data owners and the privacy of the data should be preserved at the same time. Distributed privacy preservation tends to securely distribute data through various parties. In many cases, multiple data owners tend to derive an aggregated result, which is partitioned through all other data owners. Each data owner may not want to share their own information with other parties; however, they want to perform a function such as a statistical method through the entire information and between all the parties [3]. Distributed privacy preservation is closely related to the secure computation field in cryptography. Pinkas discusses the intersection between privacy preservation in data mining and secure computation using cryptography in [47].

Yao [64] introduced secure computation for two parties, which is based on representing the secure function as a Boolean circuit. At first Yao introduced a solution to the Millionaires' Problem. The Millionaires' problem is as follows. Suppose Alice and Bob are two millionaires who want to decide which one is richer than the other without revealing their capitals to each other. Alice and Bob have  $i$  million and  $j$  million dollars, respectively where  $0 < i, j < m$  for definiteness where  $m$  is based on the memory limitation of the machine used for the computation. The solution proposed by Yao is as follows:

1. Alice establishes a public key cryptosystem where the public key is chosen ran-

domly from  $N$ -bit nonnegative integers and sends the public key to Bob.

2. Bob creates a random integer  $x$  and encrypts this value using Alice's public key  $k = E(x)$ . Bob computes  $v = k - j + 1$  and sends the result to Alice.
3. Alice computes the values of  $Y_u = D(k - j + u)$  for all  $1 \leq u \leq 10$ .
4. Alice generates a random prime integer  $p$  where  $p$  has  $N/2$  bits and computes  $Z_u = Y_u \bmod p$  for all  $1 \leq u \leq 10$ .
5. If all the values of  $Z_u$  differ by at least 2, then we go to Step 6. Otherwise Alice generates another random and repeats all stages.
6. Alice sends  $p$  and  $Z_1, Z_2, \dots, Z_i$  followed by  $Z_i + 1, Z_{i+1} + 1, Z_{i+2} + 1, \dots, Z_{10} + 1$  to Bob.
7. Bob computes  $t = x \bmod p$ . If the  $j$ -th number is equal to  $t$ , then  $i \geq j$ , otherwise  $i < j$ .

Yao introduced a model for the general problem based on the above approach [64]. The proposed approach was efficient enough for medium sized circuits, but a problem exists for large datasets, because there are some functions which could not be represented as small circuits. For example, two parties want to perform a  $k^{\text{th}}$ -ranked element function on their dataset such as median, which includes large sets of unique items. The rank depends on the size of the datasets. The size of a circuit for computing the  $k^{\text{th}}$ -ranked element would be linear in a best case scenario, which might be extremely large. Goldreich, Micali and Wigderson [29] generalized Yao's approach for multiple parties for semi-honest adversaries and then proposed a general compiler

from semi-honest to malicious. The problem with this approach was weak performance. For every multiplication gate, an oblivious transfer (a protocol in which the sender only sends a part of the information to the receiver and the receiver has no knowledge about what part has been sent to them) should have been run by the system. It also needs a communication round between all the parties for every multiplication gate. However, there was not much research completed in secure computation field until 2000.

Lindell and Pinkas [37] proposed a method to run a data mining algorithm on databases of two parties without revealing any confidential and unnecessary information, which was more efficient than generic protocols. In their work, they proposed a privacy-preserving protocol for ID3, a well-known algorithm of decision tree learning. As we discussed before, because of the lack of accuracy and security, anonymization and randomization could not be reliable when precise results are needed. As a result, much research has been done in secure computation field using cryptographic algorithms to generate a final result which is more precise and secure. However, many of the proposed methods suffer from a lack of efficiency. Cryptographic techniques are mainly used in secure computation when the data is distributed among different parties who want to perform a function securely using their private inputs. To achieve an ideal secure computation method using cryptographic algorithms, secure building blocks are needed to perform complex statistical analysis methods. Building blocks such as multiplication and summation are needed for different steps of statistical methods. Secure sum and secure multiplication are the two main important building blocks for secure computation of statistical analysis methods, although they are not the only ones. Secure comparison, secure dot product and secure count are some

other examples of secure building blocks. Clifton [13] proposed a simple and efficient method in which all parties were able to perform summation between their private data without disclosing it to others. In this protocol, one party initiates the process. The initiator generates a random number  $R$  and adds it to her private number  $N_1$  to set a new number ( $S_1 = N_1 + R$ ) which is sent to the second party. Second party adds her own private value to  $S_1$  ( $S_2 = N_2 + S_1$ ) and sends it to the third party. This procedure will be repeated until the sum  $S_n$  is received by the initiator. The initiator deducts the random number from  $S_n$  to calculate the actual sum value of the data and broadcast it to the other parties. There are some security vulnerabilities on this secure sum protocol. Two parties can compromise and reveal another party's private number (e.g. second and forth parties compromise to reveal third party's private number). Also, a malicious party can share an incorrect value, and as a result none of the parties will receive the correct value except for the malicious party. These is-

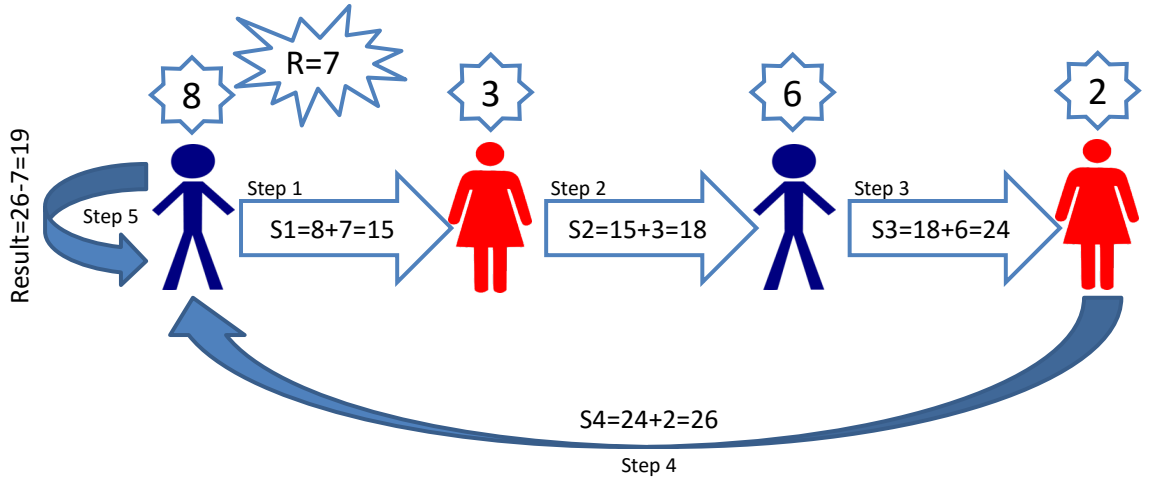


Figure 2: Clifton secure sum protocol

sues have been solved by other researchers such as Karr [34], though the new method uses a centralized server, which is a bottleneck for the system. Chaum et al. [12] proposed multiple secure protocols such as multiplication and enhanced by many other researchers. As mentioned before, we mainly follow the proposed approaches by Samet such as [52, 53, 50, 51] which includes various secure computation protocols on horizontally and vertically partitioned data.

# Chapter 4

## Proposed Approach

To this chapter, we will discuss proposed approach to enhance security and privacy in multi-party computation.

### 4.1 Data Distribution

Most of the time, data is distributed among different data custodians such as hospitals. Because of security and privacy concerns, data custodians will not allow data users (researchers) to execute a query on their databases directly. However, they are willing to perform statistical methods without revealing their private information. As shown in Figure 3, data custodians are willing to collaborate using privacy-preserving distributed algorithms to perform such methods. Therefore, secure computation is needed for running a statistical method on all records. For example, each hospital in St. John's has a number of patients, and assume that a data user wants to run a statistical method through all hospitals in St. John's. It would be the same as a data custodian distributes data through various hospitals. From this point, we refer to

above distribution as **Subset Data Distribution**. The second type of distribution

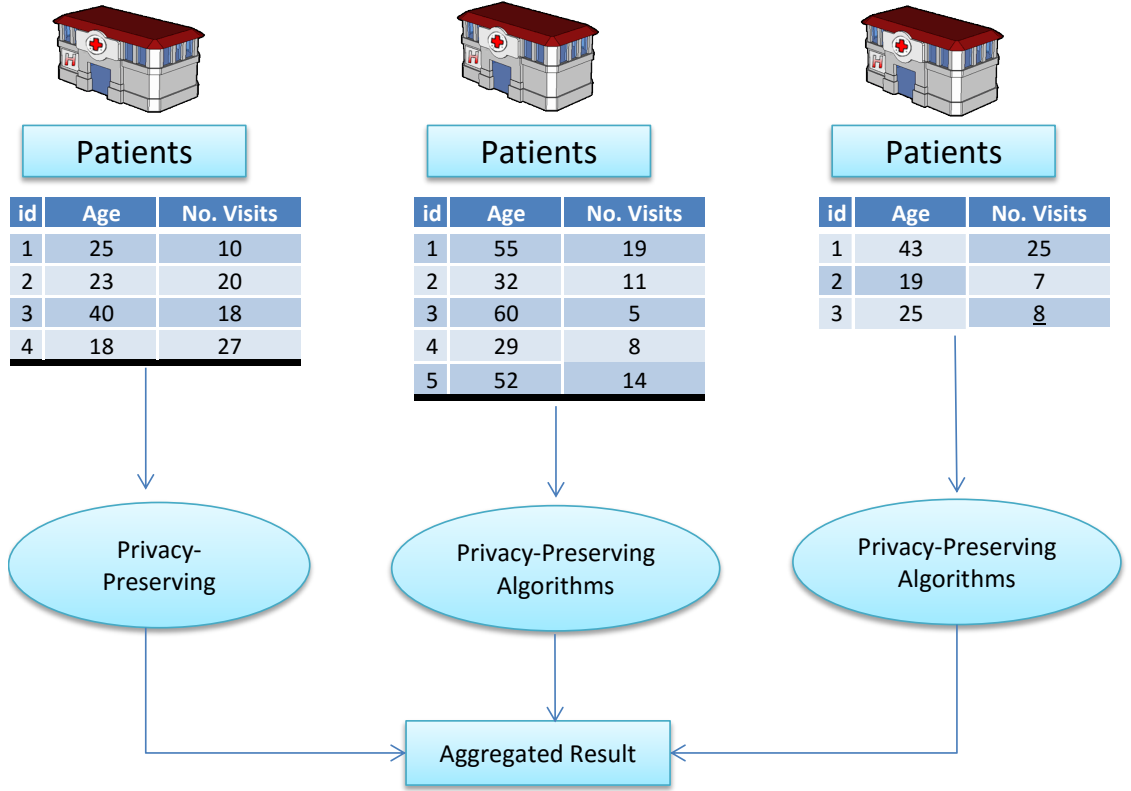


Figure 3: subset data distribution

is a more complex process, and is used when the data custodians tend to store their data in various data storages without revealing any information. As shown in Figure 4, each data value has been broken into  $n$  parts in a way that the summation of all broken parts form the original data, where  $n$  is the number of data storages. For example, imagine a data custodian wants to distribute the value of a specific column such as *Age* of a specific patient whose age is 31 through three different data storages. Data custodian breaks 31 to three different segments such as 10, 15, and 6, and

distributes these values to data storages. It can be easily seen that the summation of three values (10, 15 and 6) is equal to the original value (31). Data is distributed between data storages in a way that none of them have any knowledge over the original data. From this point, we refer to above distribution type as **Partial Data Distribution**. In subset data distribution, data may be distributed through many hospitals and through various networks, which would result in loss of performance. In partial data distribution, there is a chance of compromise between data storages to reach the exact value of private information. There exists another type of data

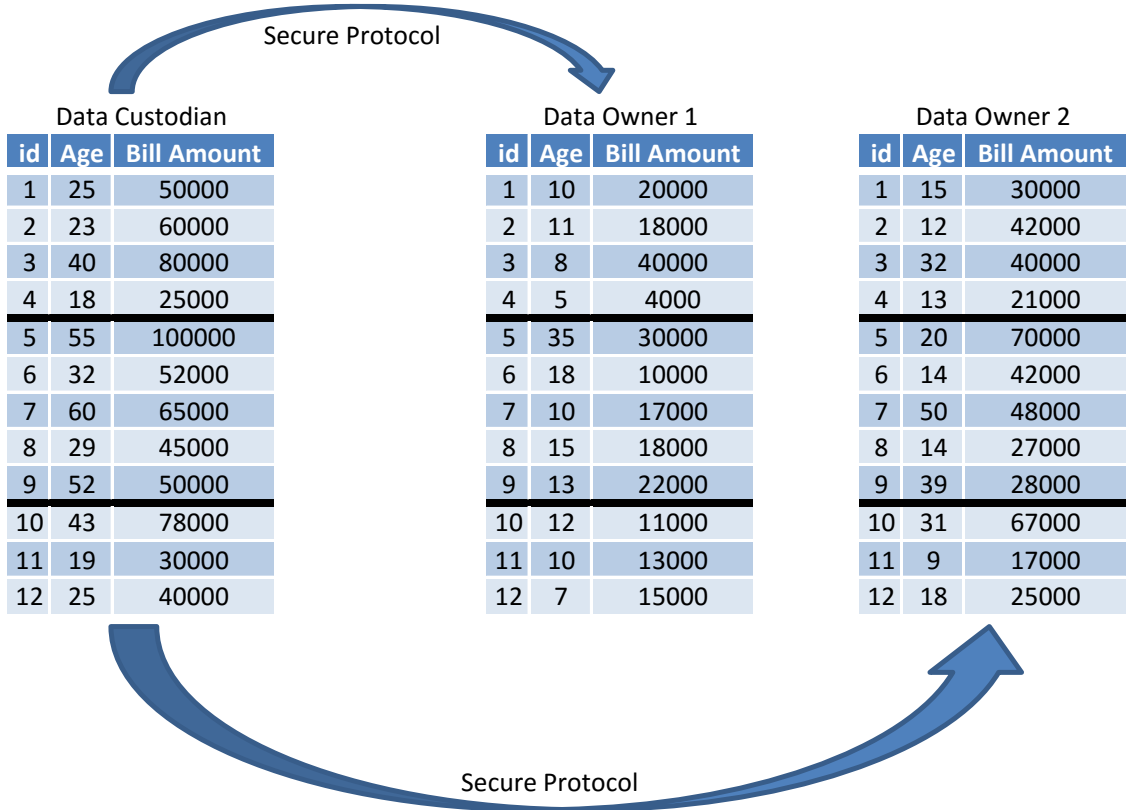


Figure 4: partial data distribution

distribution. Each party holds specific features of the data and they have the same number of records and each row in their dataset represent the same record in all parties with different features. As an example, we imagine the whole dataset is the data custodian’s dataset in Figure 3. There are two parties. The first party, holds only the first feature (e.g. Age) and the identifier’s column and the second party holds the second feature (e.g. Bill Amount) and the identifier’s column of the data custodian’s dataset. This data is distributed vertically which means that the whole dataset is a union of all vertically partitioned datasets. In this thesis, this type of data distribution is not covered and has not been implemented in the final web based framework. The reader is encouraged to look into some of the works that have been done on vertically partitioned data such as various works by Vaidya and Clifton or Samet and Miri. Some of Clifton and Vaidya works include privacy-preserving association rule mining [59], privacy-preserving k-means clustering [60], privacy-preserving naive Bayes classifier [61] and privacy-preserving decision trees [62] over vertically partitioned data. Some of Samet and Miri works include k-means clustering over vertically and horizontally partitioned data [55] and secure comparison protocol in the same paper, privacy-preserving back-propagation and extreme learning machine algorithms over horizontally and vertically partitioned data [54] and two secure protocols for perceptron learning algorithm in neural networks [52] over horizontally and vertically partitioned data.

## 4.2 Secure Building Blocks

For achieving secure computation, we use Paillier [46] cryptosystem, which is an additive homomorphic encryption system. Paillier cryptosystems does not support multiplicative homomorphism, thus the secure multiplication method proposed by Samet and Miri [53] is used for secure computation. Suppose two parties  $P_1, P_2$  want to perform secure multiplication with their own private data ( $x_{P_1}$  and  $x_{P_2}$  respectively).  $P_1$  establishes a Paillier cryptosystem and shares her public key with  $P_2$ .  $P_1$  encrypts her own private data and sends it to  $P_2$ .

$$P_1 : E(x_{P_1}) \rightarrow P_2 \quad (4.1)$$

$P_2$  generates a random private output  $y_{P_2}$ , encrypts the negative of  $y_{P_2}$  ( $-y_{P_2}$ ), and multiplies it to the encrypted data sent by  $P_1$  to the power of her own private data  $x_{P_2}$  and sends the result to  $P_1$ .

$$P_2 : E(x_{P_1})^{x_{P_2}} * E(-y_{P_2}) \rightarrow P_1 \quad (4.2)$$

$P_1$  decrypts the value given by  $P_2$ , and sets it as her own private output  $y_{P_1}$ . It can be easily shown that:

$$x_{P_1} * x_{P_2} = y_{P_1} + y_{P_2} \quad (4.3)$$

Now, suppose the two parties want to perform a secure addition with their own private data.  $P_1$  establishes a Paillier cryptosystem, and shares her public key with  $P_2$ .  $P_1$  encrypts her own private data, and sends it to  $P_2$ .

$$P_1 : E(x_{P_1}) \rightarrow P_2 \quad (4.4)$$

$P_2$  generates a random private output  $y_{P_2}$ , and multiplies the encrypted data sent by  $P_1$  to the encrypted value of her own private input and calculate the result to the power of modular inverse of  $y_{P_2}$  ( $y_{P_2}^{-1}$ ) and sends the result to  $P_1$ .

$$P_2 : (E(x_{P_1}) * E(x_{P_2}))^{y_{P_2}^{-1}} \rightarrow P_1 \quad (4.5)$$

$P_1$  decrypts the value given by  $P_2$ , and sets it as her own private output  $y_{P_1}$ . It can be easily shown that:

$$x_{P_1} + x_{P_2} = y_{P_1} * y_{P_2} \quad (4.6)$$

Now, suppose the two parties want to perform a secure dot product [28] with their own private data (we assume that both parties want to perform secure dot product on their  $k$  number of records).  $P_1$  establishes a Paillier cryptosystem, and shares her public key with  $P_2$ .  $P_1$  encrypts all  $k$  records, and sends it to  $P_2$ .

$$P_1 : E(x_1), E(x_2), \dots, E(x_k) \rightarrow P_2 \quad (4.7)$$

$P_2$  calculates the given encrypted record to the power of her own corresponding record for each set of data and calculates the multiplication of all calculated records.

$$P_2 : S = \prod_{i=1}^k E(x_i)^{y_i} \quad (4.8)$$

Afterwards,  $P_2$  generates a random private output  $y_{P_2}$ , and multiplies  $S$  to modular inverse of  $y_{P_2}$  and sends the result to  $P_1$ .

$$P_2 : S * y_{P_2}^{-1} \rightarrow P_1 \quad (4.9)$$

$P_1$  decrypts the value given by  $P_2$ , and sets it as her own private output  $y_{P_1}$ . It can be easily shown that:

$$\prod_{i=1}^k x_i y_i = y_{P_1} + y_{P_2} \quad (4.10)$$

As shown below, we can extend secure multiplication for multiple parties when there are more than two parties involved.

We assume we have  $x_1 * x_2 = y_1 + y_2$ . Thus:

$$x_1 * x_2 * x_3 = (y_1 + y_2) * x_3 = y_1 x_3 + y_2 x_3, \quad (4.11)$$

We perform two secure two-party multiplications for  $y_1 x_3$  and  $y_2 x_3$ .

$$y_1 * x_3 = z_1 + z_{31}, y_2 * x_3 = z_2 + z_{32} => \quad (4.12)$$

$$x_1 * x_2 * x_3 = z_1 + z_{31} + z_2 + z_{32} = z_1 + z_2 + z_3 \quad (4.13)$$

We assume  $x_1 * x_2 * \dots * x_i = y_1 + y_2 + \dots + y_i$ .

For  $(i + 1)$ :

$$\begin{aligned} x_1 * x_2 * \dots * x_i * x_{i+1} &= (y_1 + y_2 + \dots + y_i) * x_{i+1} \\ &= y_1 x_{i+1} + y_2 x_{i+1} + \dots + y_i x_{i+1}, \end{aligned} \quad (4.14)$$

We perform  $i$  secure two-party multiplications, so we have:

$$\begin{aligned} x_1 * x_2 * \dots * x_i * x_{i+1} &= z_1 + z_{(i+1)1} + z_2 + z_{(i+1)2} + \dots + z_i + z_{(i+1)i} \\ &= z_1 + z_2 + \dots + z_i + z_{(i+1)} \end{aligned} \quad (4.15)$$

With a similar approach, we can extend the secure addition protocol for more than two parties.

Now that we have our secure building blocks, we will show the protocols for popular statistical analysis methods. We assume that data user wants to perform any statistical method between two parties for partial data distribution and subset data distribution. As shown above, we can easily extend the methods to work on any multi-party network.

### 4.3 Statistical Analysis Methods

In this section, we will discuss the secure protocols for the computation of various statistical analysis methods. As shown in Figure 5, the data user receives metadata about data owners' records. Then, she sends her preferred query to the data owners. Data owners will perform required secure operations based on the query and each of them will send the private output to the data user. By receiving those partial information, the data user will be able to calculate the final result based on the given data. Although, the mentioned process would be similar for both distribution types, protocols for secure statistical analysis methods may differ. In this thesis, we will discuss secure mean protocol as an example and the reader is referred to [51] for further secure statistical methods analysis. In secure mean protocol, the data user receives the metadata from data owners and sends the name of the attribute (e.g. Age), which she wants to perform the secure mean protocol on to the first data owner  $D_1$ . Note that  $Age = (a_1, a_2, \dots, a_n) = (a_{1,1}, a_{1,2}, \dots, a_{1,n_1}, a_{2,1}, a_{2,2}, \dots, a_{2,n_2})$ , such that  $a_{i,n_i} \in D_i$  and  $n = n_1 + n_2$ .

$$\mu = \frac{\sum_{i=1}^n a_i}{n} = \frac{\sum_{i=1}^{n_1} a_{1,i} + \sum_{i=1}^{n_2} a_{2,i}}{n_1 + n_2} \quad (4.16)$$

After sending the query to the data owners, the process differs for subset or partial data distributions. For subset data distribution,  $D_1$  establishes the encryption keys, and sends the public keys to the second data owner  $D_2$ . Then,  $D_1$  and  $D_2$  locally calculate the summation of the corresponding column's values such that:

$$A_i = \sum_{j=1}^{n_i} a_{i,j} \quad (4.17)$$

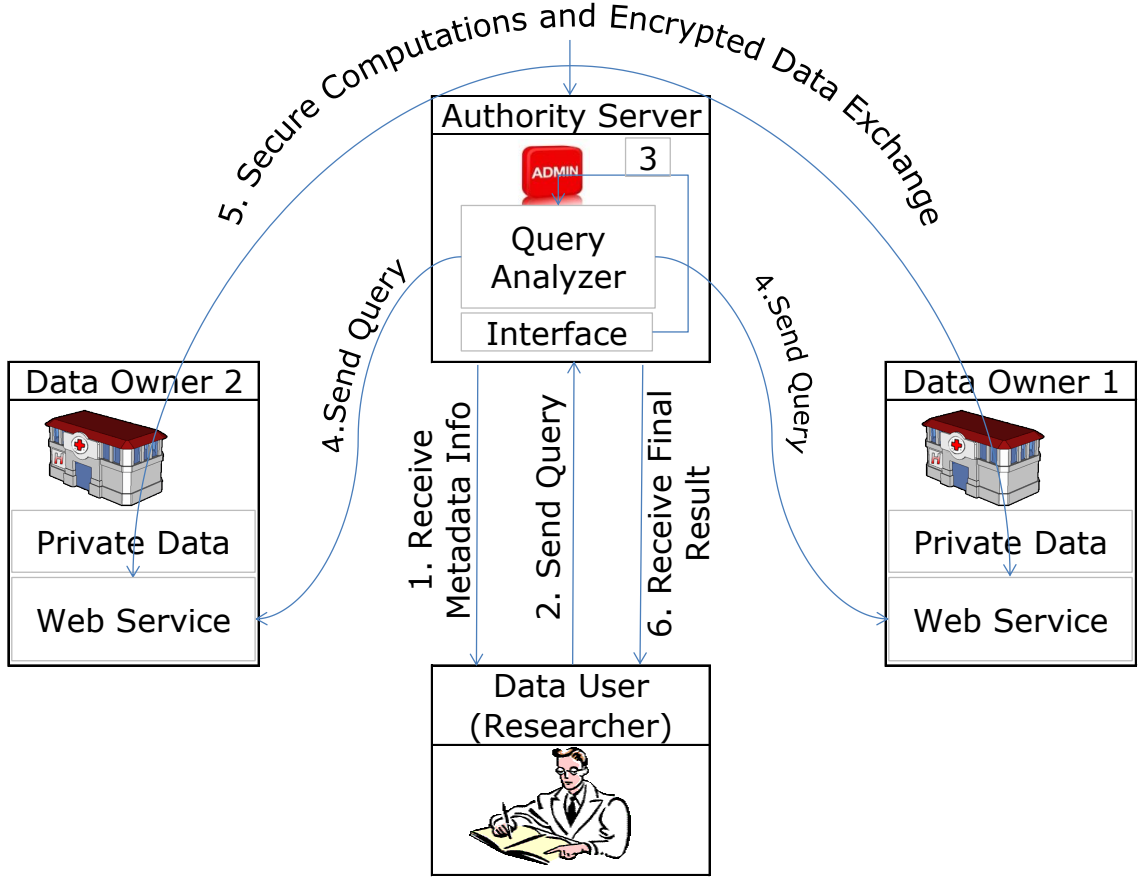


Figure 5: Orocess Flow of the Protocol

Afterwards, data owners perform a secure addition for their private values  $A_1$  and  $A_2$ . Thus, each data owner has their own private output for the summation of their values  $PO1_{D_1}$  and  $PO1_{D_2}$  where:

$$A_1 + A_2 = PO1_{D_1} * PO1_{D_2} \quad (4.18)$$

Each data owner has a number of records,  $n_1$  and  $n_2$  respectively. They perform secure summation for their records, and as a result they will have another private output  $PO2_{D_1}$  and  $PO2_{D_2}$  where  $n_1 + n_2 = PO2_{D_1} * PO2_{D_2}$ . Each data owner will send their private outputs to the data user. Data user will then compute the multiplication of

the given data for each set of outputs ( $PO1_{D_1} * PO1_{D_2}$  and  $PO2_{D_1} * PO2_{D_2}$ ). By doing so, data user has the summation of all values of the requested column and also the summation of the records. By dividing these two values, the data user has the final result such that:

$$\mu = \frac{PO1_{D_1} * PO1_{D_2}}{PO2_{D_1} * PO2_{D_2}} \quad (4.19)$$

For partial data distribution, there is no need for the parties to perform secure summation for their number of records, i.e. they have the same number of records ( $n_1 = n_2 = n$ ). Thus, one of the parties sends the number of records to the data user. The data user will multiply the given private outputs, and divide it by number of records in order to get the final value of mean, such that:

$$\mu = \frac{PO1_{D_1} * PO1_{D_2}}{n} \quad (4.20)$$

Other protocols include Secure Variance, Secure Skewness, Secure Correlation, and Secure Chi-Square Test, which are proposed by Samet et al. [51]. In this thesis, we implemented all these protocols in a web-based common framework and also we extended the work by implementing more secure statistical methods such as student's t-test, linear regression and logistic regression for both data distribution types. The proposed secure protocols for student's t-test and linear regression for subset data distribution and partial data distribution are completely original and also a new approach has been proposed for secure logistic regression for partial data distribution.

#### 4.3.1 Secure Student *t*-test

The *t*-test determines whether two sets of data are statistically different from each other. To be able to use *t*-test for secure computation, the algorithm should be broken

for two-party computation. The  $t$ -test formula is as below:

$$t = \frac{\mu_A - \mu_B}{\sqrt{var_A/n_A + var_B/n_B}} \quad (4.21)$$

where  $A$  and  $B$  are 2 sets of data,  $\mu_A$  and  $\mu_B$  are mean values of data sets  $A$  and  $B$  respectively,  $var_A$  and  $var_B$  are variance values of data sets  $A$  and  $B$  respectively and  $n_A$  and  $n_B$  are the number of records of data sets  $A$  and  $B$  respectively. Thus, for performing secure  $t$ -test protocol, two secure mean operations, two secure variance operations and two secure count operations are needed for each one of the two sets of data. The process is similar for partial data distribution and subset data distribution. Note that the secure mean and secure variance protocols are different for partial and subset data distribution. For calculating the mean and variance in partial data distribution, there is no need for secure addition for computing the number of records, because each party holds the same number of records as the original data owner. The only difference is, both parties have portion of all values and each value can be computed by adding partial values.

### 4.3.2 Secure Linear Regression

Simple linear regression determines the relationship between two sets of data and is used for assessing the association between two variables. Like other secure protocols, the linear regression formula should be broken for two-party computation. Simple

linear regression formula is as below:

$$y = a + bx \quad (4.22)$$

$$b = \frac{N\Sigma XY - (\Sigma X)(\Sigma Y)}{N\Sigma X^2 - (\Sigma X)^2} \quad (4.23)$$

$$a = \frac{\Sigma Y - b\Sigma X}{N} \quad (4.24)$$

where  $x$  and  $y$  are the variables,  $b$  is the slope of the regression line,  $a$  is the intercept point of the regression line and the y axis,  $N$  is the number of records,  $X$  is the first variable and  $Y$  is the second variable.

For subset data distribution we know that:

$$N = n_1 + n_2 \quad (4.25)$$

where  $n_1$  is the number of records of the first party and  $n_2$  is the number of records of the second party. Thus,

$$\sum_{i=1}^N X_i = \sum_{i=1}^{n_1} x_{1i} + \sum_{i=1}^{n_2} x_{2i} \quad (4.26)$$

$$\sum_{i=1}^N Y_i = \sum_{i=1}^{n_1} y_{1i} + \sum_{i=1}^{n_2} y_{2i} \quad (4.27)$$

$$\sum_{i=1}^N X_i Y_i = \sum_{i=1}^{n_1} x_{1i} y_{1i} + \sum_{i=1}^{n_2} x_{2i} y_{2i} \quad (4.28)$$

$$\sum_{i=1}^N X_i^2 = \sum_{i=1}^{n_1} x_{1i}^2 + \sum_{i=1}^{n_2} x_{2i}^2 \quad (4.29)$$

where  $X_i$  is the  $i$ -th record of first variable,  $Y_i$  is the  $i$ -th record of second variable,  $x_{1i}$  is the  $i$ -th record of first variable of first party,  $x_{2i}$  is the  $i$ -th record of first variable of second party,  $y_{1i}$  is the  $i$ -th record of second variable of first party and  $y_{2i}$  is the  $i$ -th record of second variable of second party. Thus for calculating the slope of the

regression line (b), five secure additions are needed  $(N, \Sigma X, \Sigma Y, \Sigma XY, \Sigma X^2)$ .

For partial data distribution, the protocol is completely different. We know that in partial data distribution, each party has the same number of records as the original data and:

$$X_i = v_i + u_i \quad (4.30)$$

$$Y_i = w_i + z_i \quad (4.31)$$

where  $v_i$  is the  $i$ -th record of first variable of first party and  $u_i$  is the  $i$ -th record of first variable of second party and the summation of corresponding values form the original data. It is quite similar for  $Y_i$ . Thus,

$$\sum_1^N X_i = \sum_1^N (v_i + u_i) = \sum_1^N v_i + \sum_1^N u_i \quad (4.32)$$

$$\sum_1^N Y_i = \sum_1^N (w_i + z_i) = \sum_1^N w_i + \sum_1^N z_i \quad (4.33)$$

$$\begin{aligned} \sum_1^N X_i Y_i &= \sum_1^N (v_i + u_i)(w_i + z_i) \\ &= \sum_1^N v_i w_i + v_i z_i + u_i w_i + u_i z_i \\ &= \sum_1^N v_i w_i + \sum_1^N v_i z_i + \sum_1^N u_i w_i + \sum_1^N u_i z_i \end{aligned} \quad (4.34)$$

$$\begin{aligned} \sum_1^N X_i^2 &= \sum_1^N (v_i + u_i)^2 \\ &= \sum_1^N v_i^2 + u_i^2 + 2v_i u_i \\ &= \sum_1^N v_i^2 + \sum_1^N u_i^2 + 2 \sum_1^N v_i u_i \end{aligned} \quad (4.35)$$

For calculating 4.32 and 4.33, two secure additions are needed. In 4.34,  $\sum_1^N v_i w_i$  and  $\sum_1^N u_i z_i$  can be computed locally, but two secure dot products are needed for calculating  $\sum_1^N v_i z_i$  and  $\sum_1^N u_i w_i$ . Thus, three additional secure additions are needed for the computation of 4.34. In 4.35,  $\sum_1^N v_i^2$  and  $\sum_1^N u_i^2$  can be computed locally, but one secure dot product is needed for calculating  $\sum_1^N v_i u_i$ . Thus, two additional secure additions are needed for the computation of 4.35. For calculating  $\sum X \sum Y$ , one secure multiplication is needed. For computing  $N \sum XY - (\sum X)(\sum Y)$ , another secure addition is needed. Also, for calculating  $N \sum X^2 - (\sum X)^2$  another additional secure addition is needed. In total, for calculating the slope of the regression line (b) in partial data distribution, three secure dot products, one secure multiplication and nine secure additions are needed.

### 4.3.3 Secure Logistic Regression

Logistic regression is a regression model that determines the relationship between one or more independent variable(s) with a dependent variable which is categorical. It is used to estimate the probability of a dependent variable based on independent variable(s). Like other statistical analysis methods, logistic regression should be broken in a way that each party calculates the private output for themselves and the final result can be calculated using private output(s) for two party computation. To estimate the parameters that best fit the data, maximum likelihood estimation is utilized. Our model is based on the work done by Samet [50]. We denote the number of independent samples by  $M$  in our dataset and a random dependent variable by  $Z$  which has only two values, 1 and 0. Based on the distinct values for independent variables, we

have  $N$  different combinations. We denote the samples with combination  $i$  by  $n_i$ . We denote a column vector with  $i$  elements which stores the number of records for  $Z$  where the value is 1 by  $y_i$ . The probability of  $Z$  being 1 for the  $i$ -th element has been shown by  $\pi_i$ . The independent variables could be shown in a matrix  $X$  which has the same number of rows as our sample dataset ( $M$ ) and  $K+1$  columns where  $K$  is the number of independent variables. Every element's value of the first column of the matrix is 1. For each independent variable, we should compute a vector  $\beta$  with  $K+1$  and an intercept. Newton-Raphson method has been used for this purpose.  $\beta$  can be computed as below:

$$\beta_z = \beta_{z-1} + [X^T W X]^{-1} X^T (y - \mu) \quad (4.36)$$

$$\mu_i = n_i \pi_i,$$

$$\pi_i = P(Z_i = 1|i),$$

where  $\beta_0$  is an initial approximation which can be based on the previous experiments on a similar dataset or it can be generated randomly and  $W$  is a diagonal  $N * N$  matrix. In matrix  $W$ , all the elements except for elements on its diagonal ( $e_i$ ) have a zero value, where  $e_i$  can be computed as below:

$$e_i = n_i \pi_i (1 - \pi_i)$$

We consider a specific threshold and the above computation will iterate until the difference between  $\beta_z$  and  $\beta_{z-1}$  is no less than the threshold. For all iterations, the

computation of  $\mu_i$  and  $w_{i,i}$  is as below:

$$\mu_i = \frac{e^{X_i * B_z}}{1 + e^{X_i * B_z}} \quad (4.37)$$

$$w_{i,i} = \frac{e^{X_i * B_z}}{(1 + e^{X_i * B_z})^2} \quad (4.38)$$

The  $X$ ,  $W$  and  $\mu$  matrices are as below:

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1K} \\ 1 & x_{21} & x_{22} & \dots & x_{2K} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{N1} & x_{N2} & \dots & x_{NK} \end{bmatrix}$$

$$W = \begin{bmatrix} n_1\pi_1(1 - \pi_1) & 0 & 0 & \dots & 0 \\ 0 & n_2\pi_2(1 - \pi_2) & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & n_N\pi_N(1 - \pi_N) \end{bmatrix}$$

$$\mu = \begin{bmatrix} n_1\pi_1 \\ n_2\pi_2 \\ \dots \\ n_N\pi_N \end{bmatrix}$$

For subset data distribution, each party has a subset of the whole data. We assume that the first party has  $j_1$  number of records and the second party has  $j_2$  number of records. Thus:

$$j_1 + j_2 = N \Rightarrow j_2 = N - j_1$$

The first party's  $X$  and  $\mu$  matrices are as below:

$$X_1 = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1K} \\ 1 & x_{21} & x_{22} & \dots & x_{2K} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{j_1 1} & x_{j_1 2} & \dots & x_{j_1 K} \end{bmatrix}$$

$$\mu_1 = \begin{bmatrix} n_1 \pi_1 \\ n_2 \pi_2 \\ \dots \\ n_{j_1} \pi_{j_1} \end{bmatrix}$$

The second party's matrices are as below:

$$X_2 = \begin{bmatrix} 1 & x_{(j_1+1)1} & x_{(j_1+1)2} & \dots & x_{(j_1+1)K} \\ 1 & x_{(j_1+2)1} & x_{(j_1+2)2} & \dots & x_{(j_1+2)K} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{N1} & x_{N2} & \dots & x_{NK} \end{bmatrix}$$

$$\mu_2 = \begin{bmatrix} n_{j_1+1} \pi_{j_1+1} \\ n_{j_1+2} \pi_{j_1+2} \\ \dots \\ n_N \pi_N \end{bmatrix}$$

The  $W$  matrix is public. For calculating 4.36, we proceed as follow:

$$X^T W = \begin{bmatrix} 1 & 1 & \dots & 1 & 1 & \dots 1 \\ x_{11} & x_{21} & \dots & x_{j_1 1} & x_{(j_1+1)1} & \dots x_{N1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x_{1K} & x_{2K} & \dots & x_{j_1 K} & x_{(j_1+1)K} & x_{NK} \end{bmatrix} \begin{bmatrix} n_1 \pi_1 (1 - \pi_1) & \dots & 0 \\ 0 & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & n_N \pi_N (1 - \pi_N) \end{bmatrix}$$

$$X^T W = \begin{bmatrix} n_1 \pi_1 (1 - \pi_1) & n_2 \pi_2 (1 - \pi_2) & \dots & n_N \pi_N (1 - \pi_N) \\ x_{11} n_1 \pi_1 (1 - \pi_1) & x_{21} n_2 \pi_2 (1 - \pi_2) & \dots & x_{N1} n_N \pi_N (1 - \pi_N) \\ \dots & \dots & \dots & \dots \\ x_{1K} n_1 \pi_1 (1 - \pi_1) & x_{2K} n_2 \pi_2 (1 - \pi_2) & \dots & x_{NK} n_N \pi_N (1 - \pi_N) \end{bmatrix}$$

$$\begin{aligned}
X^T W X &= \begin{bmatrix} n_1 \pi_1 (1 - \pi_1) & \dots & n_N \pi_N (1 - \pi_N) \\ \dots & \dots & \dots \\ x_{j_1+1} n_{j_1+1} \pi_{j_1+1} (1 - \pi_{j_1+1}) & \dots & x_{N1} n_N \pi_N (1 - \pi_N) \\ \dots & \dots & \dots \\ x_{1K} n_1 \pi_1 (1 - \pi_1) & \dots & x_{NK} n_N \pi_N (1 - \pi_N) \end{bmatrix} \begin{bmatrix} 1 & \dots & x_{1K} \\ \dots & \dots & \dots \\ 1 & \dots & x_{j_1 K} \\ \dots & \dots & \dots \\ 1 & \dots & x_{NK} \end{bmatrix} \\
&= \begin{bmatrix} n_1 \pi_1 (1 - \pi_1) + \dots + n_N \pi_N (1 - \pi_N) & \dots & n_1 \pi_1 (1 - \pi_1) x_{1K} + \dots + n_N \pi_N (1 - \pi_N) x_{NK} \\ \dots & \dots & \dots \\ x_{j_1+1} n_{j_1+1} \pi_{j_1+1} (1 - \pi_{j_1+1}) + \dots + x_{N1} n_N \pi_N (1 - \pi_N) & \dots & x_{j_1+1} n_{j_1+1} \pi_{j_1+1} (1 - \pi_{j_1+1}) x_{1K} + \dots + x_{N1} n_N \pi_N (1 - \pi_N) x_{NK} \\ \dots & \dots & \dots \\ x_{1K} n_1 \pi_1 (1 - \pi_1) + \dots + x_{NK} n_N \pi_N (1 - \pi_N) & \dots & x_{1K} n_1 \pi_1 (1 - \pi_1) x_{1K} + \dots + x_{NK} n_N \pi_N (1 - \pi_N) x_{NK} \end{bmatrix}
\end{aligned}$$

The first  $j_1$  summation in each cell can be done by the first party and the second party can calculate the rest. Thus, the above matrix can be broken into the summation of two matrices which can be calculated by each party separately, such that:

$$X^T W X = M_1 + M_2 \quad (4.39)$$

where  $M_1$  and  $M_2$  are the corresponding matrices of the first party and second party respectively. In order to fully compute 4.36, we should be able to securely compute the inverse of a matrix. For computing the inverse of a matrix, we use a method proposed by Han et al. [31] in which two parties can compute their private shares as below:

$$[M_1 + M_2]^{-1} = V_1 + V_2 \quad (4.40)$$

Thus, to calculate  $[X^T W X]^{-1}$ , one secure matrix inverse computation is needed. For calculating  $X^T(y - \mu)$ , two secure matrix multiplications ( $X^T y$  and  $X^T \mu$ ) and one secure addition ( $X^T y - X^T \mu$ ) are needed. Secure multiplication of matrices for partial data distribution has been shown in [50] such that:

$$P * Q = R_1 + R_2 \quad (4.41)$$

Where  $P$  and  $Q$  are the two matrices of the first party and the second party respectively. Thus, we have:

$$X^T y - X^T \mu = W_1 + W_2 \quad (4.42)$$

$$[X^T W X]^{-1} = V_1 + V_2 \quad (4.43)$$

$$\Rightarrow [X^T W X]^{-1} X^T (y - \mu) = (V_1 + V_2)(W_1 + W_2) = V_1 W_1 + V_1 W_2 + V_2 W_1 + V_2 W_2 \quad (4.44)$$

$V_1 W_1$  and  $V_2 W_2$  can be computed locally by the first party and the second party respectively. Moreover,  $V_1 W_2$  and  $V_2 W_1$  can be computed using secure matrix multiplication. For more in-depth details, reader is encouraged to read the original paper [50].

For partial data distribution, each party has  $N$  rows of records, but the original value can be formed by the summation of all broken values as mentioned before. In order to fully compute 4.36, we should be able to securely compute the inverse of a matrix. Thus, to calculate  $[X^T W X]^{-1}$  in partial data distribution, two secure matrix multiplications and one secure matrix inverse computation is needed. For calculating  $X^T (y - \mu)$ , two secure matrix multiplications ( $X^T y$  and  $X^T \mu$ ) and one secure addition ( $X^T y - X^T \mu$ ) are needed.

## 4.4 Complexity Assumptions

As mentioned before, the proposed secure computation framework is based on Paillier Cryptosystem [46]. The security relies on the security of discrete logarithms which are known to be hard to solve. There is no efficient algorithm to solve the worst case scenario for discrete logarithm problems and it can be shown that the average case's

complexity is quite the same as the worst case scenario. With major improvement and usage of cloud computing and powerful computer systems, 2048-bit key is recommended for using in cryptosystems.

#### **4.4.1 Building Blocks Complexity**

Complexity analysis of all secure building blocks have been shown below. We assume that the Paillier cryptosystem has been established by one of the data owners. The public key exchange stage between parties through authority server has not been considered in the communication cost, because it will only be executed once for every protocol. The calculation of computation costs for various building blocks are based on previous works, but the calculation of communication costs is done by the author because the communication costs are dependent on the proposed system.

##### **4.4.1.1 Secure Multiplication**

There are two encryptions and one decryption in two-party secure multiplication. By assuming that the cost of encryption and decryption are equal, we have:

$$\text{Computation cost} = 3a$$

where  $a$  is the cost of encryption and decryption.

Data owners are not connected together directly in the proposed system; however, they are able to exchange data through authority server. For two-party secure multiplication, four message are exchanged between data owners and authority server.

Thus:

$$\text{Communication cost} = 4b$$

where  $b$  is the number of bits exchanged for each message.

To improve the performance of multi-party secure multiplication of  $n$  data owners, data owners could be divided into two groups, i.e.  $P_1, P_2, \dots, P_{\lfloor \frac{n}{2} \rfloor}$  and  $P_{\lfloor \frac{n}{2} \rfloor + 1}, \dots, P_n$  are considered the first group and the second group. By recursively dividing the new groups each time until all groups are a pair of two parties, the overall time would improve significantly. So, by doing above, there are  $\frac{n(n+1)}{2}$  encryptions and  $\frac{n(n-1)}{2}$  decryptions for this building block. Thus:

$$\text{Computation cost} = \left( \frac{n(n+1)}{2} + \frac{n(n-1)}{2} \right) a = n^2 a$$

We saw that in the two-party secure multiplication the cost is  $4b$ . By denoting this cost for  $n$  parties as  $C(n)$  where  $n \geq 3$ , we have:

$$\text{Communication cost} = C(n) = C(\lfloor \frac{n}{2} \rfloor) + C(\lceil \frac{n}{2} \rceil) + (\lfloor \frac{n}{2} \rfloor * (\lceil \frac{n}{2} \rceil + 1))b \quad (4.45)$$

#### 4.4.1.2 Secure Addition

There are two encryptions and one decryption in two-party secure multiplication. By assuming that the cost of encryption and decryption are equal, we have:

$$\text{Computation cost} = 3a$$

where  $a$  is the cost of encryption and decryption.

For two-party secure addition, four messages are exchanged between data owners and authority server. Thus:

$$\text{Communication cost} = 4b$$

where  $b$  is the number of bits exchanged for each message.

There are  $\frac{(n-1)(n+2)}{2}$  encryptions and  $\frac{n(n-1)}{2}$  decryptions for  $n$  party secure addition.

Thus:

$$\text{Computation cost} = \left( \frac{(n-1)(n+2)}{2} + \frac{n(n-1)}{2} \right) a = (n-1)(n+1)a$$

All messages are sent from data owners to authority server and then from authority server to other data owners. Therefore,

$$\text{Communication cost} = (n-1)(n+2)b \quad (4.46)$$

#### 4.4.1.3 Secure Dot Product

If the secure dot product performed on  $k$  records, there are  $k+1$  encryptions and one decryption in two-party secure dot product. By assuming that the cost of encryption and decryption are equal, we have:

$$\text{Computation cost} = (k+2)a$$

where  $a$  is the cost of encryption and decryption.

For two-party secure dot product, four messages are exchanged between data owners and authority server. Thus:

$$\text{Communication cost} = 4b$$

where  $b$  is the number of bits exchanged for each message.

#### 4.4.2 Secure Statistical Analysis Methods Complexity

Complexity analysis of all secure protocols are shown below. We assume that the Paillier cryptosystem has been established by one of the data owners. As mentioned before, the complexity analysis for secure student t-test and linear regression for both data distribution types and the complexity analysis for secure logistic regression for partial data distribution are proposed by the author. We show the costs of all protocols using the calculated costs of secure building blocks as follow:

Secure multi-party multiplication computation cost :  $SMMC(n)$

Secure multi-party multiplication communication cost :  $SMMM(n)$

Secure multi-party addition computation cost :  $SMAC(n)$

Secure multi-party addition communication cost :  $SMAM(n)$

Secure multi-party dot product computation cost :  $SMDC(n)$

Secure multi-party dot product communication cost :  $SMDM(n)$

##### 4.4.2.1 Secure Mean

For subset data distribution, there are two secure additions needed to perform secure mean protocol, thus:

$$\text{Computation cost} = 2SMAC(n)$$

$$\text{Communication cost} = 2SMAM(n)$$

For partial data distribution, there is only one secure addition needed to perform secure mean protocol, thus:

$$\text{Computation cost} = SMAC(n)$$

$$\text{Communication cost} = SMAM(n)$$

#### 4.4.2.2 Secure Variance

For subset data distribution, there are two secure additions needed and one secure mean (two additional secure additions) to perform secure variance protocol, thus:

$$\text{Computation cost} = 4SMAC(n)$$

$$\text{Communication cost} = 4SMAM(n)$$

For partial data distribution, there are five secure additions needed to perform secure variance protocol, thus:

$$\text{Computation cost} = 5SMAC(n)$$

$$\text{Communication cost} = 5SMAM(n)$$

#### 4.4.2.3 Secure Skewness

For subset data distribution, there are two secure additions needed and one secure variance (two additional secure additions), as well as one secure mean (two additional

secure additions) to perform secure skewness protocol, thus:

$$\text{Computation cost} = 6SMAC(n)$$

$$\text{Communication cost} = 6SMAM(n)$$

For partial data distribution, there are five secure additions needed to perform secure skewness protocol, thus:

$$\text{Computation cost} = 5SMAC(n)$$

$$\text{Communication cost} = 5SMAM(n)$$

#### **4.4.2.4 Secure Correlation**

For subset data distribution, there are two secure additions needed and two secure variances (four additional secure additions), two secure means (four additional secure additions) to perform secure correlation protocol, thus:

$$\text{Computation cost} = 10SMAC(n)$$

$$\text{Communication cost} = 10SMAM(n)$$

For partial data distribution, there are nine secure additions needed to perform secure correlation protocol, thus:

$$\text{Computation cost} = 9SMAC(n)$$

$$\text{Communication cost} = 9SMAM(n)$$

#### 4.4.2.5 Secure Student's *t*-test

For subset data distribution, there are two secure additions needed and two secure variances (four additional secure additions), two secure means (four additional secure additions) to perform secure Student's *t*-test protocol, thus:

$$\text{Computation cost} = 10SMAC(n)$$

$$\text{Communication cost} = 10SMAM(n)$$

For partial data distribution, there are nine secure additions needed to perform secure student's *t*-test protocol, thus:

$$\text{Computation cost} = 9SMAC(n)$$

$$\text{Communication cost} = 9SMAM(n)$$

#### 4.4.2.6 Secure Linear Regression

For subset data distribution, there are five secure additions needed to perform secure linear regression protocol, thus:

$$\text{Computation cost} = 5SMAC(n)$$

$$\text{Communication cost} = 5SMAM(n)$$

As mentioned before, for partial data distribution, there are five secure additions and three secure dot products needed to perform secure linear regression protocol, thus:

$$\text{Computation cost} = 9SMAC(n) + 3SMDC(n) + SMMC$$

$$\text{Communication cost} = 9SMAM(n) + 3SMDM(n) + SMMM$$

#### 4.4.2.7 Secure Logistic Regression

We denote the computation cost of secure matrix multiplication and secure matrix inversion by  $SMTC$  and  $SMIC$  respectively. Also, we denote the communication cost of secure matrix multiplication and secure matrix inversion by  $SMTM$  and  $SMIM$  respectively. For partial data distribution, we have two secure matrix multiplications and one secure matrix inversion, thus:

$$\text{Computation cost} = 2SMTC(n) + SMIC(n)$$

$$\text{Communication cost} = 2SMTM(n) + SMIM(n)$$

For partial data distribution, we have six secure matrix multiplications and one secure matrix inversion, thus:

$$\text{Computation cost} = 6SMTC(n) + SMIC(n)$$

$$\text{Communication cost} = 6SMTM(n) + SMIM(n)$$

## 4.5 Statistical Analysis Methods in Healthcare

Statistical analysis methods have various usage in healthcare. In this section, we present various example uses of statistical analysis methods used in healthcare studies. For example, for getting average age of patients who have chronic disease between 2002 and 2006 in Newfoundland, the statistical mean can be used. For getting the number of specific kind of patients (i.e. those who suffer from diabetes) in a certain city or district, a count method can be used. For explaining variation between actual costs and planned costs in healthcare organizations, the variance method can be used [19]. In most healthcare organizations, skewed data is an issue in order to calculate the costs. For overcoming this issue, skewness method can be used for calculating costs in healthcare [39]. Analyzing the distribution of risk-factor-attributable disease burden is an important factor to decrease the costs for a healthcare organizations and skewness is one of the most important statistical analysis methods for this purpose which was used in [49]. For understanding a lot more about the relationship between daily hassles, uplifts and major life events to health status, correlation analysis can be used [16]. Student's  $t$ -test has been used for describing the attitude of medical students towards patient-centered care considering individual preferences and param-

eters [30]. Diabetes knowledge test has two important components: a 14-item general test and a 9-item insulin-use subscale. A diabetes knowledge test has been done on two different populations using  $t$ -test [25]. Dr. Hancox [32] used linear regression to examine the relationship between television viewing and health indicators such as serum cholesterol and body-mass index. Also, in the same research, the relationship between current smoking of patients of age 26 and television viewing has been examined using logistic regression. For assessing the risk of a new drugs in a market, data must be gathered through various sources. To analyze the risk and reduce the risks of the new drugs, logistic regression should be used. A secure distributed multi-party computation protocol for logistic regression has been developed for adverse drug reaction and surveillance [20]. For monitoring the effectiveness of HPV vaccination various data registries are required and to link all these data from different registries, odds ratio is needed. A secure protocol has been proposed to address this issue [21]. These are just some limited examples of wide usage of statistical analysis methods in healthcare. For describing and examining various aspects of health and the relationship between parameters in health and healthcare, statistical analysis methods are used.

# Chapter 5

## Implementation

In this thesis, I have implemented a web-based secure multi-party computation system using Java Servlet framework for the backend programming language, and I used HTML, CSS and Javascript as frontend programming languages. Also, MySQL has been used as our database management server.

Our web application has three different types of users: data user, data owner and administrator. As shown in Figure 6, each type of user interacts with the system in various ways.

As shown in Figure 7, the data user sends the query and the query analyzer validates the query. If the query is not valid, the query analyzer displays an error message. Otherwise, the query analyzer sends the query to the repositories for collecting data. Afterwards, the collected data will be sent to the secure computation protocol, which is the main part of the implementation. Then, the secure computation protocol returns various outputs and sends them to corresponding repositories. The repositories send the output to the data user, who will compute the final result from the given

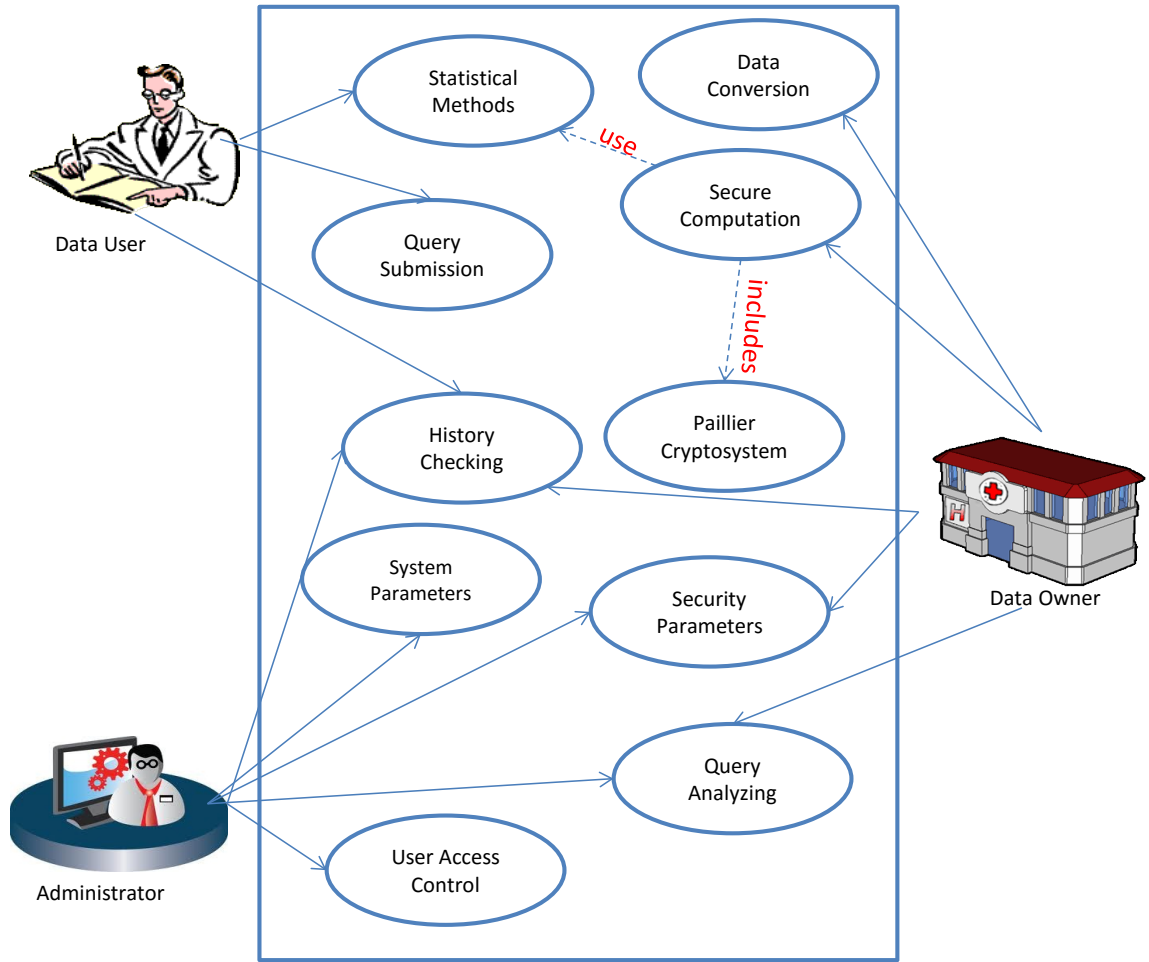


Figure 6: Use Case Diagram

inputs. Thus, different panels have been considered for each type of user. Data users would be able to login to their own panel using their own credentials, which is confirmed by the administrator. Data users can view a list of metadata that is provided by the data owners and can send a query to data owners through system administrator. For a better user experience, a graphical query builder has been implemented, such that users can create their own queries with zero knowledge of SQL queries. For example, they would be able to send any query by selecting the preferred attribute

(column) and the condition of their query from a dropdown list. Data owners would be able to select their preferred level of security, change their data conversion configurations, block unsafe queries and view their queries and records' history. The administrator would be able to define system parameters and security level, create data users and data owners or approve them upon registration, manage data transfer between data owners and data users and view the history of the system. To be able

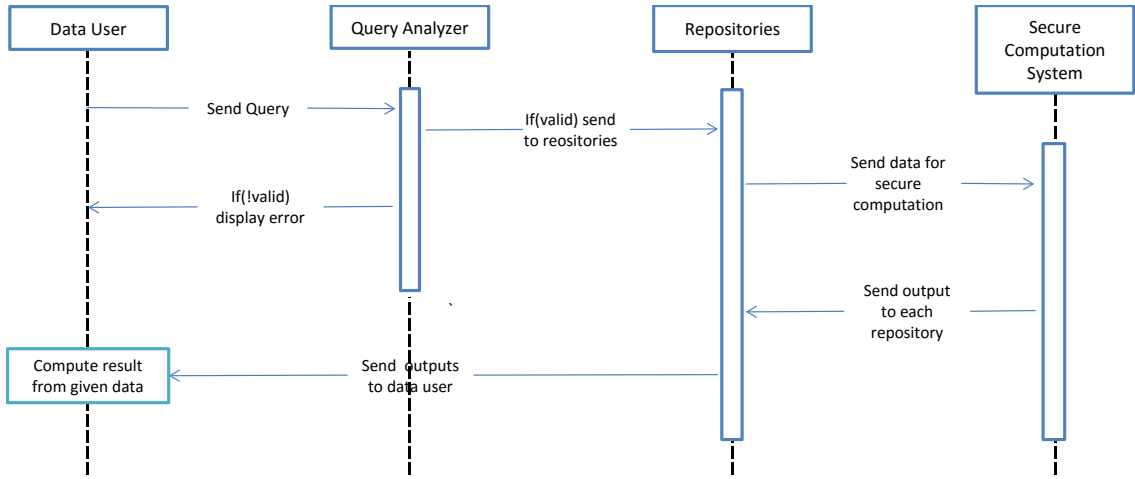


Figure 7: Sequence Diagram

to find out more about the proposed methods and the expected results (which are covered in Chapter 6), we have implemented a simple desktop version of the secure multi-party computations mentioned in this thesis, which was originally proposed by Samet et al. [51]. We have used Java programming language for implementation and RMI module is used for data transfer between the parties. As shown in Figure 8, the data user sends the attribute and desired statistical method to the data owners directly and receives the result while being able to view the process flow. The

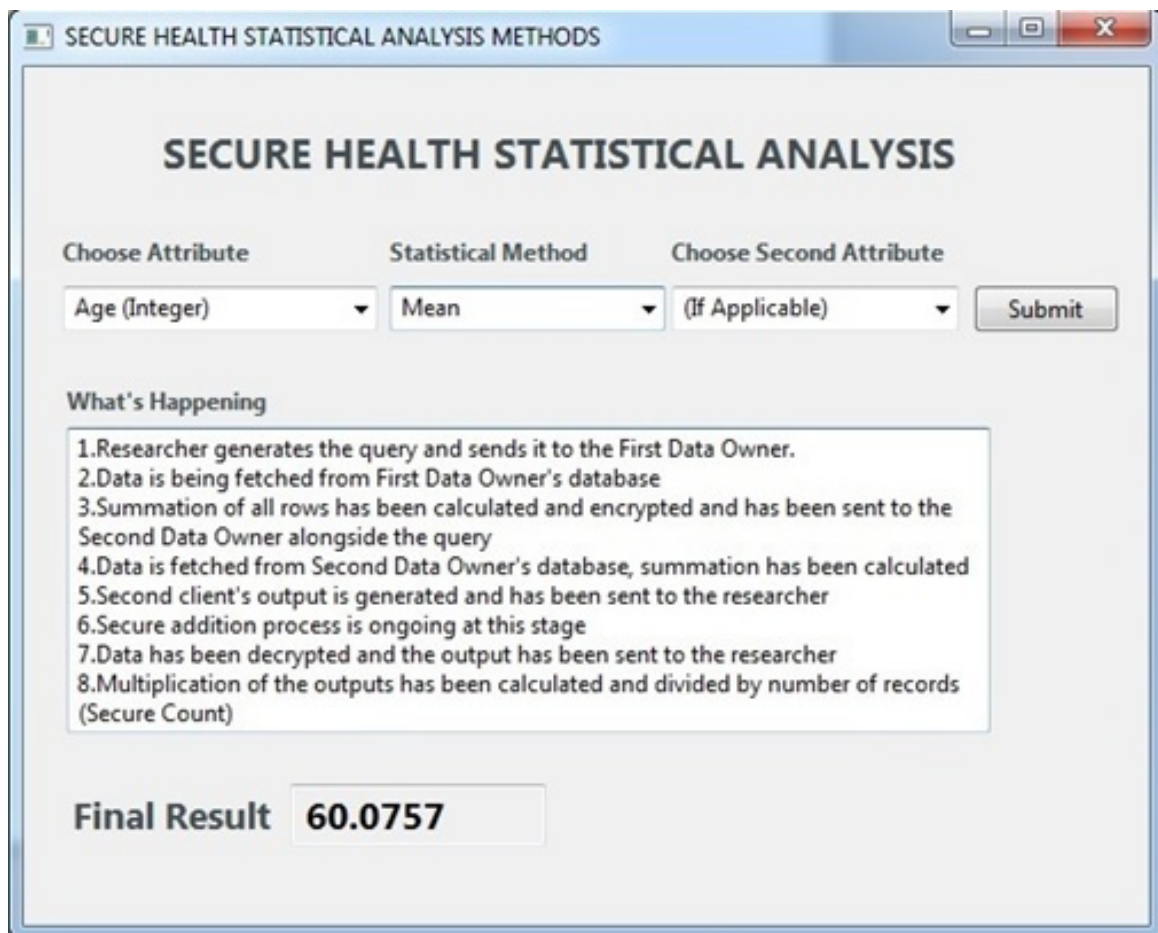


Figure 8: Desktop Application

framework follows MVC (Model-View-Controller) [35] software architectural pattern which is one of the most popular software design patterns. Models are Java classes to manage the data directly and are completely independent from user interface. Views are HTML or JSP pages for showing the representation of the data and the users will interact with the system through these pages. Controllers are mostly JSP pages to update both views and models and help transfer input and output between views and models. A controller updates the view when the model changes and also updates the model when the user interacts with the system through view.

The web framework consists of two major dependent components: The core and the web service. Most of the functionalities of the website such as authentication, registration, history view and query submitting lie in the core component. Also, all the views and the user interface are handled and shown in this component. The web service only runs on data owners' systems. Data owners communicate with the centralized server using their web services. For example, when a query is submitted by a researcher the query is sent to data owners through their web services and the web service decides to invoke a certain method and returns the result for further use.

## 5.1 Java Classes

The implementation consists of various Java classes for managing processes in the system.

### 5.1.1 Database Helper Class (DBHelper.java)

For working with the database, a Database Helper class has been implemented to simplify the process. Every interaction with the database would go through this class. It consists of various methods to implement the four basic database functions (Create, Read, Update and Delete). Repositories (see Figure 7) mostly use this class for fetching data from their local databases. Below are the methods for Database Helper class and their usage.

**insert:** This method is implemented for inserting a single row into a database table. It gets the name of the table, an array of columns' names, an array of values for each cell of any data type, the name of the primary column and the

value of the primary column as an input. It will return the *id* of the insertion if the insert is successful. If the new row contains a duplicate value of the primary column of the given table, the method returns  $-1$  which means the new row has not been inserted because of an existing same value for the primary column. If there is a problem with the database such as connecting to the database, the method returns 0.

**insertBatch:** This method is implemented for inserting multiple rows into a database table. It gets the name of the table, an array of columns' names and an array of values for each cell of any data type as an input. It will return 1 if the insert is successful. If there is a problem with the database such as connecting to the database, the method returns 0.

**update:** This method is implemented for updating row(s) of a table. It gets the name of the table, an array of columns' names, an array of values for each cell of any data type and a string of condition(s) as an input. It will return 1 if the update is successful. Otherwise, it will return 0 which means there is a problem with the database such as connecting to the database or the database server is offline.

**delete:** This method is implemented for deleting row(s) of a table. It gets the name of the table and a string of condition(s) as an input. It will return 1 if the delete is successful. Otherwise, it will return 0 which means there is a problem with the database such as connecting to the database or the database server is offline.

**select:** This method is implemented for retrieving row(s) from a single table. This is the most frequent method that is used in the implemented framework. It gets the name of the table, an array of columns' names and a string of condition as an input. It returns a *ResultSet* Java data type which contains the retrieving rows if the query is successful. Otherwise, it returns *0* if the query is unsuccessful.

**selectLeftJoin:** This method is implemented for retrieving row(s) from a table joined by a related table when the given table contains a column related to another table. This method is used many times in the project and it is one of the most frequently used kinds of queries. It gets the name of the table, an array of columns' names, the name of the second table, the name of the corresponding column of the first table, the name of the corresponding column of the second table and a string of condition as an input. It returns a *ResultSet* Java data type which contains the retrieving rows if the query is successful. Otherwise, it returns *0* if the query is unsuccessful.

**customSelect:** This method is implemented for retrieving row(s) from table(s) using a custom query. Sometimes the query for retrieving row(s) from a database is too complicated that none of the mentioned select queries could achieve it. Thus a custom select query method is implemented for complex select queries. It gets the query as an input. It returns a *ResultSet* Java data type which contains the retrieved rows if the query is successful. Otherwise, it returns *0* if the query is unsuccessful.

### 5.1.2 Global Variables Class (Globals.java)

For storing global variables, a global variables class has been implemented. Sometimes various Java classes need to access pre-defined static variable(s) for their usage such as the root directory of the project.

### 5.1.3 System Messages Class (Message.java)

Any software system shows various messages to its users such as errors which could be a meaningful message to the user, an error code or any non-human readable message. For example as shown in Figure 7) would display an error message to data user by using this class. In this implementation, a system message class has been implemented for returning meaningful messages to the user which are generated by the system. Also, this class could be useful for multilingual purposes and depending on the language selection of the user, the corresponding message to the chosen language can be returned. It consists of a single *get* method which gets a message code string as an input and returns the corresponding meaningful message.

### 5.1.4 Utility Class (Utility.java)

This class contains general methods that can be used by multiple classes. Below are the methods for Utility class and their usage.

**sha256:** This method is implemented for hashing a string using known *sha256* cryptographic hash function. It gets a string as an input. It returns the corresponding hash string if the string is not *null*. Otherwise, it returns *null*.

**convertToBigInt:** This is a series of methods for converting various Java data types such as String, int, double and timestamp(date) to BigInteger for calculation purposes. It gets one of the mentioned data types as an input and returns the corresponding BigInteger value.

**convertBigIntTo:** This is a series of methods for converting Java BigInteger data type to their original data type and their values. It gets a BigInteger value as an input and returns the corresponding original value.

**createRandomKey:** This method is implemented for generating a random unique string mostly for performing secure protocols and returns a String which uses Java built-in *randomUUID* function for generating random strings.

**sqrt:** This method is implemented for calculating the square root of a BigInteger (or BigDecimal) number. It gets a BigDecimal value as an input and returns the square root of the value.

### 5.1.5 Convert Class (Convert.java)

This class is used for getting all the values and their data types from a database table and converting various Java data types to BigInteger using Utility Class *convertToBigInt* method and store them in another database table for further calculations. This method is used for pre-processing stage and is used when the data changes or based on a defined schedule (i.e. every Saturday). The *convert* method gets the name of the original table, name of the converted table and a list of column names (including their data types) as an input. It reads all the data from the original table and

converts them to BigInteger data type and stores them in the given converted table. Repositories (see Figure 7) mostly use this class for secure computation purposes.

### 5.1.6 User Class (User.java)

One of the most used classes in this framework is the User class. This class provides the necessary basic methods for users of the system from creating a new user in the system to checking if a user has access to a single page. Users can have their email address as their username or choose a unique name for their username. The *noUsername* property defines their preference. If *noUsername* is true, the given email will be considered as their username. Below are the methods for User class and their usage.

**getId:** This method returns the *id* property of a *User*'s object.

**getUsername:** This method returns the *username* property of a *User*'s object.

**getEmail:** This method returns the *email* property of a *User*'s object.

**getType:** This method returns the *type* property of a *User*'s object which can be *a* for administrator, *o* for data owner and *r* for a researcher.

**getAccess:** This method returns the access property of a *User*'s object which can be *16* for administrator, *8* for data owner and *4* for a researcher.

**getActivate:** This method returns the *activate* property of a *User*'s object which can be *1* if the user is activated or *0* if the user is not activated.

**getCreateDate:** This method returns the *createDate* property of a *User*'s object which contains the date in which the user has been created.

**createUser:** This method is implemented for creating (registering) a new user into the system. It gets the *username*, *password*, *email*, *type*, *access code* and *activate* value as an input. Note that before sending the data to the database helper class, this method hashes the password for security reasons. If the user has been created successfully, it returns a *Pair* Java data type which contains the insertion code returned by database helper class and a successful user registration code. If the username or email address exists in the database, it returns a *Pair* which contains the insertion code and a user existence message code. Otherwise, it returns a *Pair* which contains the insertion code and a database error code.

**updateUserById:** This method is implemented for updating a specific user's information. It gets the *id*, *password*, *type*, *access code* and *activate* value of the user as an input. Note that *email* and *username* cannot be changed in this framework. If the user has been created successfully, it returns a *Pair* Java data type which contains the update code returned by database helper class and a successful user registration message code. Otherwise, it returns a *Pair* which contains the update code and a database error code.

**getUsers:** This method is implemented for retrieving all users and their properties from the database. It returns an array of *Users*' object if the query is executed successfully. Otherwise, it returns an empty array of *Users* object.

**getUserById:** This method is implemented for retrieving a specific user and their properties from the database. It gets the *id* property of the preferred user as an input and returns a *User* object if the query is executed successfully. Otherwise, it returns an empty *User* object.

**deleteUserById:** This method is implemented for deleting a specific user from the database. It gets the *id* property of the preferred user as an input and returns a *Pair* containing the delete code returned by database helper class and a successful user deletion message code if the query is executed successfully. Otherwise, it returns a *Pair* which contains the delete code and a database error code.

**createProfile:** This method is implemented for creating a new user profile in the system. Each user can have various attributes such as *name* and *address* and this method will create a profile for a single user. It gets the *id*, *first name*, *last name*, *phone* and *address* of the user as an input. If the profile has been created successfully, it returns a *Pair* Java data type which contains the insertion code returned by database helper class and a successful profile creation message code. If the *id* of the user exists in the database, it returns a *Pair* which contains the insertion code and a user existence message code. Otherwise, it returns a *Pair* which contains the insertion code and a database error code.

**activateUserById:** This method is used for activating a single user. For security reasons, when a new researcher or data owner has been created, their accounts should be activated by the administrator. This method gets the *id* of the user as an input and returns a *Pair* containing the update code returned by

database helper class and a successful user activation message code if the query is executed successfully. Otherwise, it returns a *Pair* which contains the delete code and a database error code.

**deactivateUserById:** This method is used for deactivating a single user. For security reasons, sometime a user can be deactivated by the administrator. This method gets the *id* of the user as an input and returns a *Pair* containing the update code returned by database helper class and a successful user deactivation message code if the query is executed successfully. Otherwise, it returns a *Pair* which contains the delete code and a database error code.

**login:** This method is used for logging in a single user into the system using their *username* and *password*. It gets the *username* and *password* as an input and after hashing the *password*, sends a query to the database to select the corresponding user using database helper class. It returns a *Pair* which contains a *User* object and a successful user login message code if the query is executed successfully and if the user exists in the database. If the user does not exist in the database given the *username* and *password*, it returns a *Pair* which contains a *null User* object and an unsuccessful user login message code. If the user has not been activated, it returns a *Pair* which contains a *null User* object and a deactivated user message code. Otherwise, it returns a *null User* object and database error code.

**checkAccess:** This method is for checking if a specific user has access to a specific page. As mentioned before, each user has an *access* attribute which could be 16, 8 and 4 for admin, data owner and researcher respectively. Also,

each page has an access value which can be stored in the database or hard coded into the page itself as a variable. In this framework, the later has been used for its simplicity. To check if a single user has access to a specific page, a logical conjunction operand would be operated on the user's *access* attribute and the page's *access* attribute and if the result is 0, the user does not have access to that specific page. Otherwise, user can access the specific page. For example, if a page's *access* attribute is 16, it means that only the administrator can access the page ( $16 \& 16 = 16, 16 \& 8 = 0, 16 \& 4 = 0$ ). For giving access to multiple types of users for a single page, the page access can be the summation of the preferred user types. For example, if the page access is 20, both administrator and researcher can access the page using the *checkAccess* method ( $20 \& 16 = 16, 20 \& 8 = 0, 20 \& 4 = 4$ ). There are three major benefits of implementing the users' access using the mentioned method. First, it is computationally efficient, because it works with integers and simple logical conjunction operator. Second, it can be easily extended for any number of user types. New user type's access property can be any integer of powers of 2 and pages' accesses can be easily modified for new user types. Finally, it can be easily used for object access in future rather than page access. Instead of giving access to pages, every object can have an *access* property and it can be easily calculated to see if a user has access to a specific object.

### 5.1.7 Paillier Class (Paillier.java)

As mentioned before, the Paillier cryptosystem has been used for performing secure computation in the framework. Thus, some of the core functionalities and methods lie in this class. The Paillier class would be used mainly for initializing the cryptosystem (key generation), encrypting and decrypting a message. Repositories (see Figure 7) mostly use this class for sending data for secure computation purposes. Below are the methods for Paillier class and their usage.

**KeyGeneration:** This method is used for generating the public keys and private keys of the system based on the given length of the key bits and the certainty of the generated prime numbers which is given as inputs and sets the public keys and private keys of the corresponding object. Also, it sets a random number that is used for encryption.

**Encryption:** This method is used for encrypting a message. In this system, all the messages are converted to BigInteger Java data type, because Paillier cryptosystem (and almost every cryptosystem) works with integer numbers. Thus, this method gets the message as an input and returns the encrypted message using the public keys generated in the key generation phase.

**Decryption:** This method is used for decrypting a message. It gets the ciphertext (a very large integer number) as an input and returns the original message using the private keys generated in the key generation phase.

### 5.1.8 Query Analyzer Class (QueryAnalyzer.java)

When a query is sent by the researcher, the Query Analyzer class will be invoked. At first, the query sent by the user will be analyzed to check if it is safe for processing. Then the query will be recorded in the database for future references. After that the query will be sent to data owners for processing using an http call to their web services. Query Analyzer (see Figure 7) mostly uses this class for verifying queries sent from data user and sending queries to repositories. Below are the methods for Query Analyzer class and their usage.

**checkQuery:** This method is used for validating the query sent by the researcher. It gets the *id* property of the user, the preferred statistical analysis methods, the columns' names and the condition of the query. It returns *true* if the query is safe, otherwise it returns *false*.

**addToHistory:** This method is used for recording a query in the database for future references. It gets the *id* property of the user, the preferred statistical analysis methods, the columns' names and the condition of the query and saves a record in the database and returns a *Pair* containing the insertion code returned by database helper class and a successful query created message code if the query is executed successfully. Otherwise, it returns a *Pair* which contains the insertion code and a database error code.

**sendQueryToDataOwner:** This method is used for sending a submitted query to a single data owner. It gets the web service URL of the data owner and the preferred parameters and sends the data using HTTP Post method and gets

the final message from the data owner and returns a *JSON* object. It returns *null* if the connection is lost.

### 5.1.9 Secure Computation Class (SecureComputation.java)

The Secure Computation class is the most important section of the implementation. It will handle all secure operations from a simple secure addition to complex statistical analysis methods such as logistic regression. When two (or more) data owners want to perform a secure statistical analysis method, they call various methods from this class. Secure Computation (see Figure 7) mostly uses this class for performing secure operations on data. Below are the methods for Secure Computation class and their usage.

**establishPaillier:** This method is used for establishing a Paillier cryptosystem between two data owners. This method is used by the first data owner and the data owner establishes a Paillier cryptosystem and sends the public keys to the second data owner. This method gets a unique key for future references as an input and returns a Java Map Object containing the public keys.

**setPublicKeys:** This method is used for establishing a Paillier cryptosystem between two data owners. This method is used by the second data owner and sets the public keys sent by the first data owner for the second data owner. This method gets a unique key for future references as an input and Paillier public keys ( $g$ ,  $n$ , *bit length*).

**secureAddition:** This method is used for performing a secure addition. We assume that the data sent to this method has already been encrypted. Thus,

this method performs an operation on the data based on which data owner calls this method. It gets the data owner's client number (first or second data owner), self-encrypted value and the value sent by the other data owner as the input. If the second data owner is performing the secure addition operation, it returns its own final output and also another output that should be decrypted by the first data owner. If the first data owner is performing secure addition, it returns the final output by decrypting the value sent by the second data owner.

**count:** This method is used for performing a secure count between two parties and it acts as a setup phase for performing the operation. It gets the preferred query condition as an input and calculates the the number of records in the converted database based on the query. Then, it encrypts the the number of records using the established cryptosystem and returns the values.

**mean:** This method is used for performing a secure mean between two parties and it acts as a setup phase for performing the operation. It gets the preferred column name and the query condition as an input and calculates the summation of all stored values and the number of records in the converted database based on the query. Then, it encrypts the summation and the number of records using the established cryptosystem and returns these two values.

**variance:** This method is used for performing a secure variance between two parties and it acts as a setup phase for performing the operation. It gets the preferred column name and the query condition as an input and calculates the summation of all stored values to the power of 2 ( $S$ ) and the number of records in the converted database based on the query. Then, it encrypts  $S$  and the

number of records using the established cryptosystem and returns these two values.

**skewness:** This method is used for performing a secure skewness between two parties and it acts as a setup phase for performing the operation. It gets the preferred column name and the query condition as an input and calculates the summation of all stored values to the power of 3 ( $S$ ) and the number of records in the converted database based on the query. Then, it encrypts  $S$  and the number of records using the established cryptosystem and returns these two values.

**correlation:** This method is used for performing a secure correlation between two parties and it acts as a setup phase for performing the operation. It gets two column names and the query condition as an input and calculates the summation of the multiplication of corresponding column values ( $S$ ) and the number of records in the converted database based on the query. Then, it encrypts  $S$  and the number of records using the established cryptosystem and returns these two values.

**linear:** This method is used for performing a secure linear regression between two parties and it acts as a setup phase for performing the operation. It gets two column names and the query condition as an input and calculates the summation of the multiplication of corresponding column values, addition of all values of the first column, addition of all values of the second column, addition of all squared values of the first column and the number of records in the converted database based on the query. Then, it encrypts all the mentioned values using

the established cryptosystem and returns these values.

**logistic:** This method is used for performing a secure logistic regression between two parties and it acts as a setup phase for performing the operation. It gets two column names and the query condition as an input and calculates the summation matrices of the multiplication matrices of corresponding column values ( $S$ ) and the number of records in the converted database based on the query. Then, it encrypts  $S$  and the number of records using the established cryptosystem and returns these two values.

#### 5.1.10 Data Exchange

The Data Exchange class handles all the data exchange between data owners and calculates the final result based on the given values by data owners. When two (or more) data owners want to perform a secure statistical analysis method, they call various methods from this class. The implemented methods for various secure protocols are similar, thus only secure mean method is described as an example. Mean method gets a unique key, preferred column name, the query's condition, a list of data owners and their web service URL and the public keys as an input. Then it sends the operation (i.e. mean), condition and the column name to data owners (for simplicity, we assume we have two data owners). Then it gets two sets of encrypted values from data owners. Then it sends the encrypted summation value from the first data owner to the second data owner. Then it requests a secure addition between two data owners and gets two private output from data owners. It follows a similar process for getting private values from data owners for the number of records. Then

it multiplies the first set of private outputs for calculating the summation of all values and also calculates the total number of records by multiplying the second set of private outputs. By dividing these two values, the average (mean) is calculated. The implemented web framework currently supports two data owners, but it can be easily extended to support multi-party secure computations. Repositories and Query Analyzer (see Figure 7) mostly use this class for exchanging data.

## 5.2 Controllers

As mentioned before, the controllers are a bridge between the models (Java classes) and views. When a user logs into the system, they will be redirected to the main page. The main page is the dashboard of the system in which every user can see their recent activities and the corresponding menu items. Important controllers are listed below.

### 5.2.1 Main Controller (index.jsp)

This controller handles which dashboard should be shown to the logged-in user based on the user type. It also redirects non-users to the login page. Moreover, it loads the header and footer section of the framework which contains the necessary Javascript and CSS libraries.

### 5.2.2 Menu Controller (leftColumn.jsp)

Based on the type of a user (administrator, data owner or researcher), the menu items differ. This controller gets the type of the user using *getType User's* class

method and loads the relevant menu items to the user. For example, administrator should have access to see the list of all users in the system and be able to activate or deactivate a user. Thus, this controller only shows the Users List menu item only to the administrator.

### 5.2.3 Login Controller (loginCheck.jsp)

This controller gets the input from the view (login.html) and calls the User's class *login* method. If the login is successful, it sets the session parameters such as *userId* and *userType* for future use and redirects the user to the main page where they can view their dashboard based on the user type. Otherwise it will display the error message and redirects the user back to the login page.

### 5.2.4 Logout Controller (logoutC.jsp)

This controller is called when a user clicks on the logout button. It invalidates the current session and destroys it and redirects the user to the login page.

### 5.2.5 Secure Computation Controller (queryC.jsp)

This controller is the most important controller of the implemented framework which handles the data exchange between data owners for performing secure statistical analysis methods. It gets the query from the user and calls the Query Analyzer's *checkQuery* and *addToHistory* methods. If the query is safe, it sends the query to data owners' web services using Query Analyzer's *sendQueryToDataOwner* method and then receives the data from various data owners and sends needed data to data owners

for performing the preferred statistical analysis method. For example, for performing a secure mean between two data owners, it sends a secure mean request to both data owners' web services and receives the encrypted value of summation and number of records from both data owners. Then it sends a secure addition request to the second data owner's web service for the summation and passes the first data owner's encrypted data to the second data owner's web service for calculations. After that, it receives the second's data owner final output and the second data owner's encrypted data (*output1*). Then it sends *output1* to the first data owner for decryption and receives the final output of the first client. It repeats this process for the number of records and receives another set of outputs. Then it calculates the aggregated result by dividing the multiplication of each set of outputs and sends the result to the view for showing to the researcher.

### 5.3 Web Service

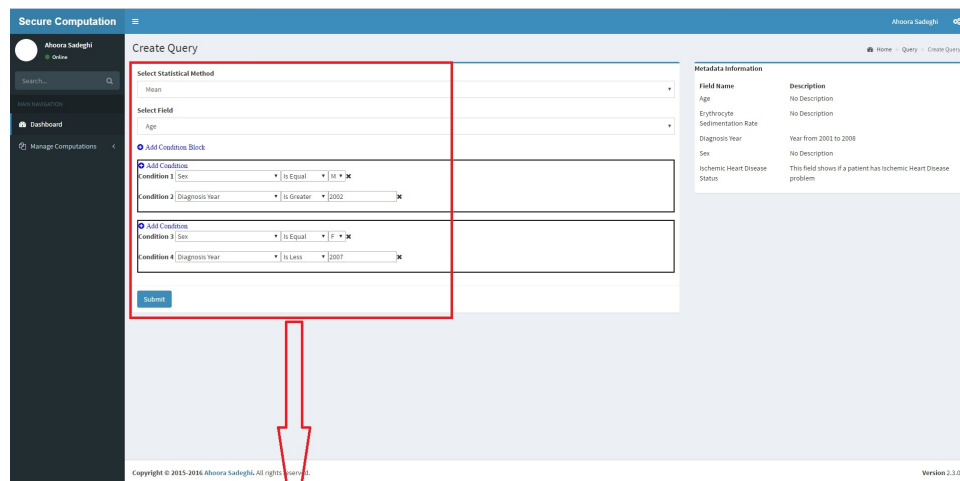
The web service runs on data owners' local computers and receives the data from the centralized server. Based on the given data, the web service decides to perform the preferred operation such as establishing a Paillier cryptosystem or performing one of the secure statistical analysis methods. It calls the corresponding method from Secure Computation class and returns the output as a string. The result will be converted to *JSON* Java Object by the centralized server.

## 5.4 Views

Administrators and data users interact with the framework through the views. For example, administrators can see the list of registered users and meta data and data users would be able to see their query history through the views. One of the most important section of the views are the query builder page. Data users are able to create their own preferred query without any knowledge of SQL queries. First, they can select their preferred statistical analysis method and the column(s) in which they want to perform their query on. Then, they can add unlimited number of conditions and condition blocks for creating their own queries. Between each condition block, there exists an OR operand and conditions in each condition blocks are connected together by an AND operand. For example, for getting the average Age of patients from data owners in which the patients are male and have been diagnosed after 2002 or the female patients who have been diagnosed before 2007 they can proceed as follow:

They can select *Mean* as preferred statistical analysis method. Then they can select *Age* as their preferred field. Two condition blocks should be added for creating the mentioned query. There should be two conditions in each condition block. The first field in the first condition should be *Sex*. The second field in the first condition should be *Is Equal* and the last field in the first condition should be *M*. The first field in the second condition should be *Diagnosis Year*. The second field in the second condition should be *Is Greater* and the last field in the second condition should be *2002*. These two conditions are stored in the first condition block. The second condition block is quite similar only with a few details changed. Figure 9 shows the screenshot that

has been taken directly from the web framework for the mentioned query. When the submit button is clicked, the result page will be shown to the user. Also, the submitted query will be shown as a human readable text. As shown in Figure 10 the average age of patients with the defined conditions is 60.30 in our dataset. For another example, we assume a researcher wants to find a relation between patients' two variables using correlation. For example, the relation between a patient's number of times that has been visited by an intern and a patient's ischemic heart disease status for patients who are older than 45 years. As shown in Figure 11, the researcher can select the two preferred fields (*Intern Visited* and *Ischemic Heart Disease Status*) and can add the preferred condition (Age Is Greater than 45) and select correlation as statistical analysis method and finally click the submit button. Figure 12 shows the result (which is  $-0.0114$ ) for the above query in our dataset. For another example, we assume a researcher wants to find a linear relation between patients' two variables using linear regression. For example, the relation between a patient's number of times that has been visited by an intern and a patient's age for patients who have ischemic heart disease. As shown in Figure 13, the researcher can select the two preferred fields (*Intern Visited* and *Age*) and can add the preferred condition (*Ischemic Heart Disease Status* equals to 1) and select linear regression as statistical analysis method and finally clicks the submit button. Figure 14 shows the result (which is  $y = 58.93 + 0.01x$ ) for the above query in our dataset.



### Select Statistical Method

Mean

### Select Field

Age

### + Add Condition Block

#### + Add Condition

Condition 1 Sex Is Equal M

Condition 2 Diagnosis Year Is Greater 2002

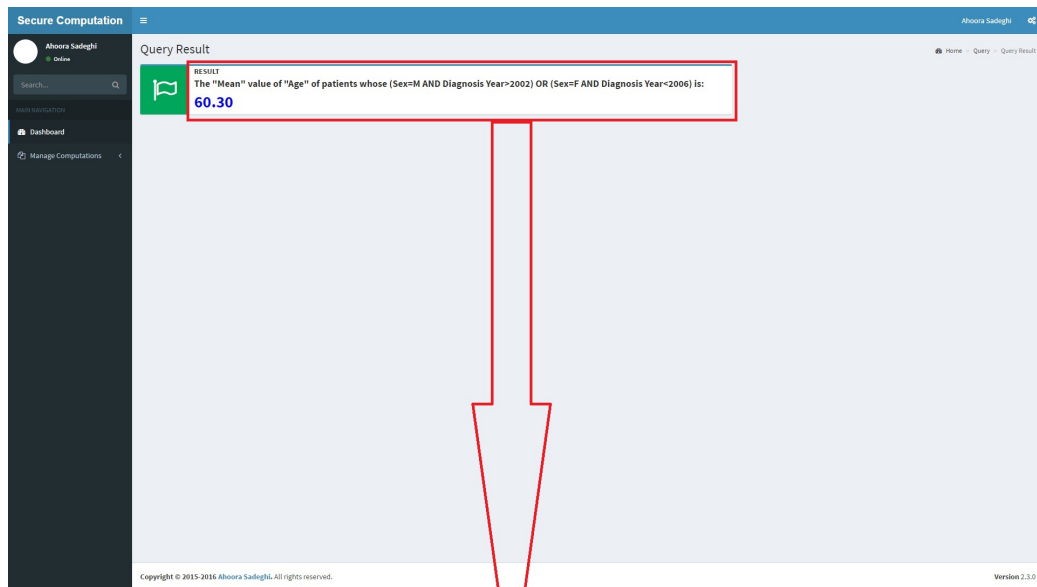
#### + Add Condition

Condition 3 Sex Is Equal F

Condition 4 Diagnosis Year Is Less 2007

Submit

Figure 9: Mean Query Page



RESULT

The "Mean" value of "Age" of patients whose (Sex=M AND Diagnosis Year>2002) OR (Sex=F AND Diagnosis Year<2006) is:

60.30

Figure 10: Mean Query Result

Secure Computation

Alhoora Sadeghi

Order

Search...

Home

Dashboard

Manage Computations

Create Query

Select Statistical Method

Correlation

Select Field

Intern Visited

Select Second Field

Ischemic Heart Disease Status

Add Condition Block

Add Condition

Condition 1 | Age | Is Greater | 45

Submit

Metadata Information

Field Name	Description
Age	This field shows the age of a patient
Erythrocyte Sedimentation Rate	This field shows the erythrocyte sedimentation rate of a patient
Diagnosis Year	Year from 2001 to 2008
Sex	This field shows the sex of a patient
Ischemic Heart Disease Status	This field shows if a patient has Ischemic heart Disease problem
Is Deceased	This field shows if a patient is deceased or not
Death Date	This field shows the death date of a patient
Intern Visited	This field shows how many times a patient has been visited by an intern

Copyright © 2015-2016 Alhoora Sadeghi. All rights reserved.

Version 2.3.0

### Select Statistical Method

Correlation

### Select Field

Intern Visited

### Select Second Field

Ischemic Heart Disease Status

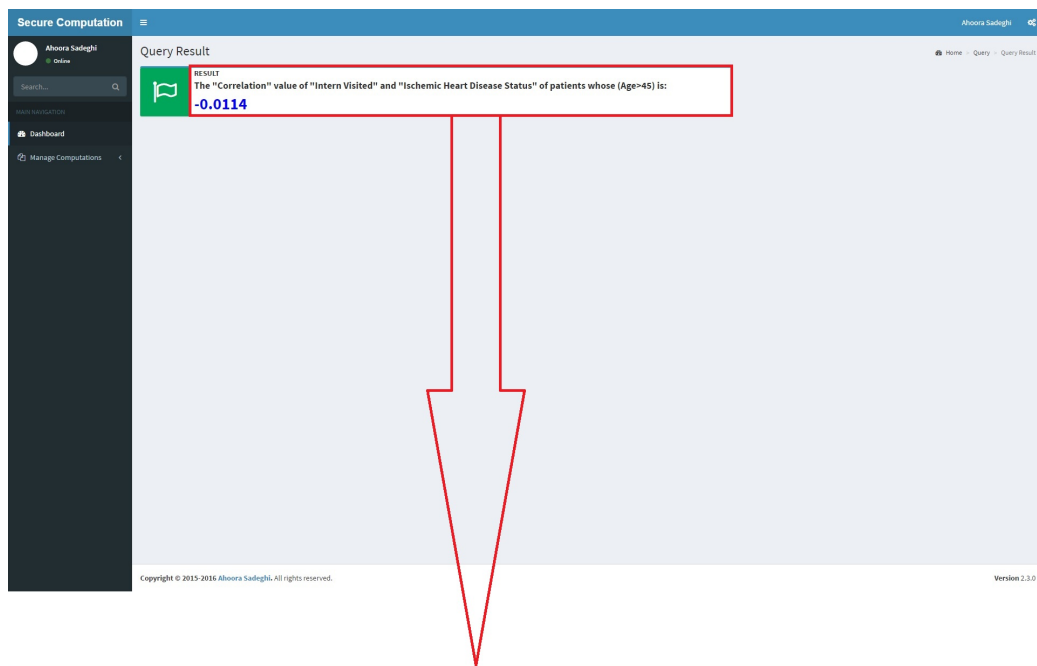
+ Add Condition Block

+ Add Condition

Condition 1 | Age | Is Greater | 45

Submit

Figure 11: Correlation Query Page

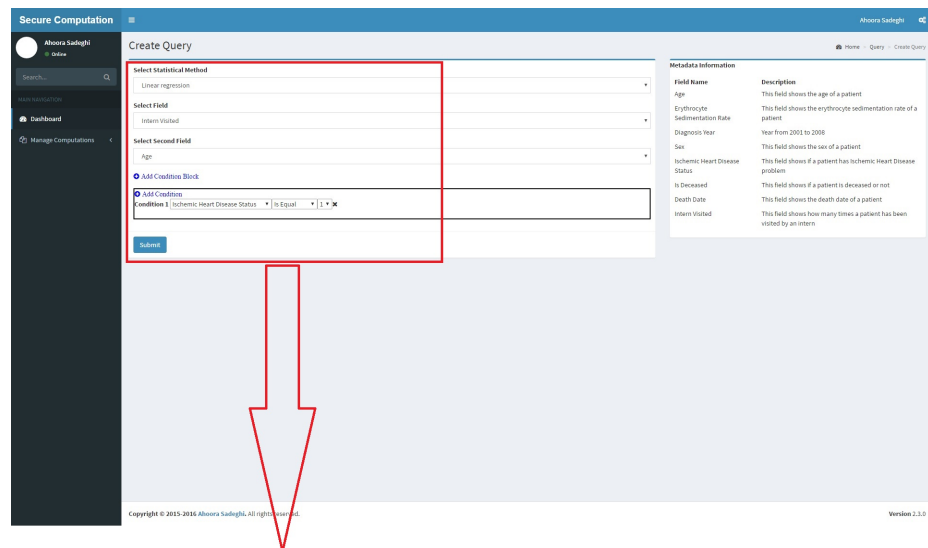


RESULT

The "Correlation" value of "Intern Visited" and "Ischemic Heart Disease Status" of patients whose (Age>45) is:

**-0.0114**

Figure 12: Correlation Query Result



### Select Statistical Method

Linear regression

### Select Field

Intern Visited

### Select Second Field

Age

### + Add Condition Block

#### + Add Condition

Condition 1 Ischemic Heart Disease Status Is Equal 1 X

Submit

Figure 13: Linear Regression Query Page

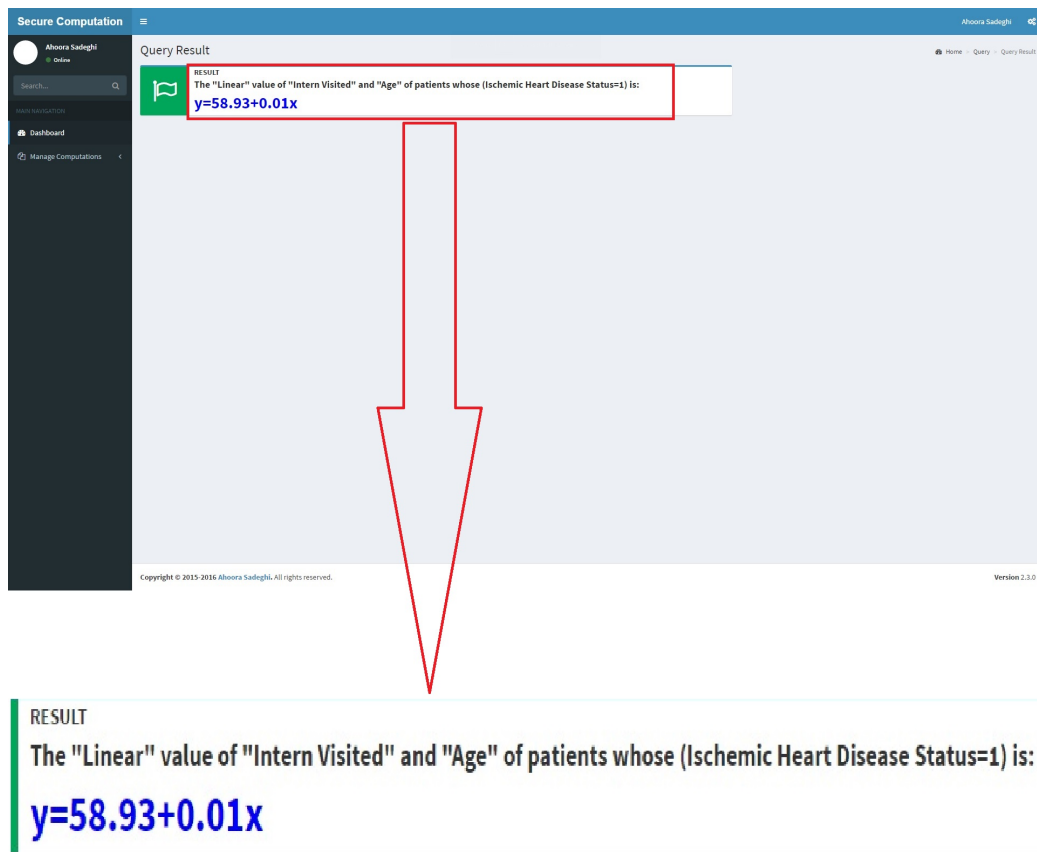


Figure 14: Linear Regression Query Result

# Chapter 6

## Results

As mentioned in Chapter 5, implementation of a simple desktop application helped us to acquire a better knowledge of the expected results. Our dataset consists of 10,000 records of data with 109 columns. To investigate the performance of the proposed protocols, we first convert all the data to BigInteger for calculation purposes, as a pre-processing stage and store the converted data in another database. This stage will only have to be executed once or when the stored data changes. Then, we implemented the secure addition building block. The hardware and software specifications for the experiment are as follows:

- Windows 7 Professional
- Intel Core i5-4570 3.20GHz
- 8 GB DDR3 RAM

Table 2 shows the performance of pre-processing stage for different data types and mixed data types. By mixed data types, we mean converting all data types at the

<b>Data Type</b>	<b>Time (in seconds)</b>
Integer	2.168
Date Time	2.949
String	2.767
Double	2.889
Mixed (for 10 different columns)	36.307

Table 2: Performance Results for the Pre-processing Stage

<b>Algorithm</b>	<b>Time (in seconds)</b>
Encryption	0.00014
Decryption	0.00980
Secure Addition	0.08726

Table 3: Performance Results for the Cryptosystem and Secure Addition

same time. It can be seen that converting the mixed data takes more time than other data types. The reason is that the framework first should detect the data type for each column and then call the corresponding method to convert the data accordingly. Table 3 shows the performance of the encryption, decryption and secure addition. The encryption key length in this experiment is 1024 bits. To achieve higher level of security, the key length could be larger, e.g. 2048 bits. Increasing the size of the key would degrade the performance of the system. The framework uses Java built-in *BigInteger* and *BigDecimal* classes for performing mathematical operations on large integer numbers. Table 4 illustrates the overall time for each protocol when the number of records is 10,000 (fetching the data from database is also included

Algorithm	Time (in seconds)
Count	0.284
Mean	0.468
Variance	0.613
Correlation	1.366
Chi-square	0.748
Skewness	0.956

Table 4: Performance Results for the Protocols Including Fetching Data from Database

in the calculated time). Table 5 illustrates the overall time for each protocol when the number of records is 10,000 if the data is stored in memory. By doing above experiments, we found out that the proposed system was practical and could be fully implemented as a web-based application and the web framework could be compared with the desktop application. The final web-based framework has been implemented using TCP protocol in order to work through different networks, which is faster than

Algorithm	Time (in seconds)
Count	0.067
Mean	0.191
Variance	0.327
Correlation	0.854
Skewness	0.575

Table 5: Performance Results for the Protocols when Data is Stored in Memory

RMI protocol. Thus, we expect improvement in performance.

## 6.1 Performance Analysis

In this section, we want to analyze the performance of the proposed protocols, building blocks and other processes (i.e. establishing the cryptosystem). The encryption and decryption in the web application are not different than those in the desktop application, because they both use the same resources of the system and these two processes are independent from exchange protocol. But the key exchange process is different, because the key is transferred through TCP protocol and RMI protocol for web and desktop application respectively. Establishing Paillier Cryptosystem and exchanging data between two data owners took 38 milliseconds. To find out more about the performance of the secure protocols, we have to consider a couple of parameters such as the complexity of the query and number of records. Four different scenarios have been considered for showing the performance of all protocols:

- Table 6 shows the performance of the protocols when no conditions have been added to the query with 10000 records.
- Table 7 shows the performance of the protocols when no conditions have been added to the query with 100000 records.
- Table 8 shows the performance of the protocols when there exists a complex

Algorithm	Time (in seconds)
Count	0.035
Mean	0.049
Variance	0.095
Skewness	0.163
Correlation	0.237
Student's $t$ -test	0.196
Linear Regression	0.079
Logistic Regression	0.823

Table 6: Secure Protocols Performances with no conditions and 10000 records

Algorithm	Time (in seconds)
Count	2.019
Mean	2.196
Variance	4.283
Skewness	6.679
Correlation	11.513
Student's $t$ -test	12.388
Linear Regression	2.396
Logistic Regression	38.207

Table 7: Secure Protocols Performances with no conditions and 100000 records

Algorithm	Time (in seconds)
Count	0.033
Mean	0.046
Variance	0.091
Skewness	0.153
Correlation	0.246
Student's <i>t</i> -test	0.217
Linear Regression	0.072
Logistic Regression	0.910

Table 8: Secure Protocols Performances with conditions and 10000 records

condition with 10000 records. The *where* condition is as below for all protocols:

Condition: Where (*Age* > 30 AND *Diagnosis\_Year* < 2007) OR (*SEX* = "M")

- Table shows the performance of the protocols when there exists a complex condition with 100000 records. The *where* condition is as below for all protocols:

Condition: Where (*Age* > 30 AND *Diagnosis\_Year* < 2007) OR (*SEX* = "M")

All the tests have been operated on *Subset Distrubted Data* and only two parties have been involved. It can be seen from the above experiments that adding complexity to

Algorithm	Time (in seconds)
Count	2.146
Mean	2.374
Variance	4.415
Skewness	6.584
Correlation	10.993
Student's $t$ -test	11.030
Linear Regression	2.230
Logistic Regression	35.418

Table 9: Secure Protocols Performances with conditions and 100000 records

the query does not affect the performance. The reason is, although adding conditions to a query make a query more complex and increases the query time, it usually reduces the number of returned records which makes up for the performance degradation. Moreover, increasing the number of records affects the protocols' performances by a larger margin. It can be seen that almost all of the protocols' processing times are 50 times more when the number of records are multiplied by 10. The main reason for this degradation is because of database queries. Performance of queries in *MYSQL* depends highly on the number of retrieved records. Also calculation on encrypted data takes more time when the number of encrypted records is increased.

In conclusion, it can be seen that the proposed system is completely practical and can be used for various purposes when the data is distributed among various data owners. All the protocols have been processed in an acceptable time on a normal computer. Also, sample records of DEPICT database have been used for running tests to test the

practicality of the system. Moreover, if more than two data owners exist in the system, the performance would decrease significantly based on the complexity assumptions which were mentioned in Chapter 4. There are many ways to improve the system which is covered in Chapter 7 such as running the final web-based framework on powerful servers in order to perform better on larger datasets.

## Chapter 7

# Conclusion and Future Work

In this work, we implemented a secure computation web-based framework based on Samet et al. [51] approach for popular statistical analysis methods used in various applications, especially in health data. Also, we proposed three more statistical analysis methods and added to the framework. In addition, various features such as an authorized server for user access control and plenty of system parameters (e.g. security level) have been added to the framework. Computations and secure data communications are done by two or more data owners on the private distributed data and only the final results will be decrypted and sent back to the data user. As a proof of concept, the final web application was tested over the Digital Epidemiology Chronic Disease Tool (DEPICT) database. This database includes health data from Canadian Chronic Disease Surveillance System between 1998 and 2004 for adults who have chronic disease in Newfoundland and Labrador.

As the future work, the web application could be extended to cover other important statistical methods in health research and other fields of science. Also, the perfor-

mance and efficiency of the final web application could be improved in various ways, such as implementing the application in multi-thread context. Also, the framework could be extended to support secure computation over vertically partitioned data for all implemented secure statistical analysis methods. Moreover, the web application could be implemented in a way that it could be easily integrated with one or more statistical analysis software programs, such as R and SAS.

# Bibliography

- [1] C. C. Aggarwal. On k-anonymity and the curse of dimensionality. In *Proceedings of the 31st international conference on Very large data bases*, pages 901–909. VLDB Endowment, 2005.
- [2] C. C. Aggarwal. On randomization, public information and the curse of dimensionality. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 136–145. IEEE, 2007.
- [3] C. C. Aggarwal and S. Y. Philip. A general survey of privacy-preserving data mining models and algorithms. In *Privacy-preserving data mining*, pages 11–52. Springer, 2008.
- [4] R. Agrawal. Data mining: Crossing the chasm. *KDD '99: Proceedings of the fifth ACM SIGKDD International Conference On Knowledge Discovery and Data Mining*, 1999.
- [5] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *ACM Sigmod Record*, volume 29, pages 439–450. ACM, 2000.
- [6] M. Anker. Epidemiological and statistical methods for rapid health assessment. *World health statistics quarterly*, 44(3), 1991.

- [7] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 217–228. IEEE, 2005.
- [8] L. L. Beck. A security mechanism for statistical database. *ACM Transactions on Database Systems (TODS)*, 5(3):316–3338, 1980.
- [9] J. M. Bland and D. G. Altman. Statistics notes: measurement error. *Bmj*, 313(7059):744, 1996.
- [10] D. Boneh, C. Gentry, S. Halevi, F. Wang, and D. J. Wu. Private database queries using somewhat homomorphic encryption. In *Applied Cryptography and Network Security*, pages 102–118. Springer, 2013.
- [11] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Advances in Cryptology—CRYPTO 2011*, pages 505–524. Springer, 2011.
- [12] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 11–19. ACM, 1988.
- [13] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *ACM Sigkdd Explorations Newsletter*, 4(2):28–34, 2002.
- [14] R. Cramer, I. Damgård, and J. B. Nielsen. *Multiparty computation from threshold homomorphic encryption*. Springer, 2001.

- [15] I. Damgård, V. Pastro, N. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology–CRYPTO 2012*, pages 643–662. Springer, 2012.
- [16] A. DeLongis, J. C. Coyne, G. Dakof, S. Folkman, and R. S. Lazarus. Relationship of daily hassles, uplifts, and major life events to health status. *Health psychology*, 1(2):119, 1982.
- [17] D. E. Denning and T. F. Lunt. A multilevel relational data model. In *Security and Privacy, 1987 IEEE Symposium on*, pages 220–220. IEEE, 1987.
- [18] J. Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism\*. In *Information security*, pages 471–483. Springer, 2002.
- [19] H. Dove and T. Forthman. Helping financial analysts communicate variance analysis. *Healthcare financial management: journal of the Healthcare Financial Management Association*, 49(4):52–54, 1995.
- [20] K. El Emam, S. Samet, L. Arbuckle, R. Tamblyn, C. Earle, and M. Kantarcioglu. A secure distributed logistic regression protocol for the detection of rare adverse drug events. *Journal of the American Medical Informatics Association*, 20(3):453–461, 2013.
- [21] K. El Emam, S. Samet, J. Hu, L. Peyton, C. Earle, G. C. Jayaraman, T. Wong, M. Kantarcioglu, F. Dankar, and A. Essex. A protocol for the secure linking of registries for hpv surveillance. *PloS one*, 7(7):e39915, 2012.

- [22] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in cryptology*, pages 10–18. Springer, 1984.
- [23] F. Faltin, R. Kenett, and F. Ruggeri. *Statistical Methods in Healthcare*. John Wiley & Sons, 2012.
- [24] S. R. A. Fisher. On the "probable error" of a coefficient of correlation deduced from a small sample. 1921.
- [25] J. T. Fitzgerald, M. M. Funnell, G. E. Hess, P. A. Barr, R. M. Anderson, R. G. Hiss, and W. K. Davis. The reliability and validity of a brief diabetes knowledge test. *Diabetes care*, 21(5):706–710, 1998.
- [26] C. Gentry et al. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, pages 169–178, 2009.
- [27] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. Fast data anonymization with low information loss. In *Proceedings of the 33rd international conference on Very large data bases*, pages 758–769. VLDB Endowment, 2007.
- [28] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen. On private scalar product computation for privacy-preserving data mining. In *International Conference on Information Security and Cryptology*, pages 104–120. Springer, 2004.
- [29] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229. ACM, 1987.

- [30] P. Haidet, J. E. Dains, D. A. Paterniti, L. Hechtel, T. Chang, E. Tseng, and J. C. Rogers. Medical student attitudes toward the doctor–patient relationship. *Medical education*, 36(6):568–574, 2002.
- [31] S. Han and W. K. Ng. Privacy-preserving linear fisher discriminant analysis. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 136–147. Springer, 2008.
- [32] R. J. Hancox, B. J. Milne, and R. Poulton. Association between child and adolescent television viewing and adult health: a longitudinal birth cohort study. *The Lancet*, 364(9430):257–262, 2004.
- [33] D. W. Hosmer Jr and S. Lemeshow. *Applied logistic regression*. John Wiley & Sons, 2004.
- [34] A. F. Karr, W. J. Fulp, F. Vera, S. S. Young, X. Lin, and J. P. Reiter. Secure, privacy-preserving analysis of distributed databases. *Technometrics*, 49(3):335–345, 2007.
- [35] G. E. Krasner, S. T. Pope, et al. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, 1(3):26–49, 1988.
- [36] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE, 2007.

- [37] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology CRYPTO 2000*, pages 36–54. Springer, 2000.
- [38] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.
- [39] A. S. Malehi, F. Pourmotahari, and K. A. Angali. Statistical models for the analysis of skewed healthcare cost data: a simulation study. *Health economics review*, 5(1):1–16, 2015.
- [40] N. Mantel. Chi-square tests with one degree of freedom; extensions of the mantel-haenszel procedure. *Journal of the American Statistical Association*, 58(303):690–700, 1963.
- [41] K. V. Mardia. Measures of multivariate skewness and kurtosis with applications. *Biometrika*, 57(3):519–530, 1970.
- [42] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 223–228. ACM, 2004.
- [43] B. Mihaylova, A. Briggs, A. O’Hagan, and S. G. Thompson. Review of statistical methods for analysing healthcare resources and costs. *Health economics*, 20(8):897–916, 2011.
- [44] M. Morgenstern. *Security and inference in multilevel database and knowledge-base systems*, volume 16. ACM, 1987.

- [45] J. Neter, M. H. Kutner, C. J. Nachtsheim, and W. Wasserman. *Applied linear statistical models*, volume 4. Irwin Chicago, 1996.
- [46] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptologyEUROCRYPT99*, pages 223–238. Springer, 1999.
- [47] B. Pinkas. Cryptographic techniques for privacy-preserving data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):12–19, 2002.
- [48] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
- [49] A. Rodgers, M. Ezzati, S. Vander Hoorn, A. D. Lopez, R.-B. Lin, C. J. Murray, et al. Distribution of major health risks: findings from the global burden of disease study. *PLoS Med*, 1(1):e27, 2004.
- [50] S. Samet. Privacy-preserving logistic regression. *Journal of Advances in Information Technology Vol*, 6(3), 2015.
- [51] S. Samet, S. Asghari, A. Sadeghi Boroujerdi, and O. Hurley. Secure health statistical analysis (shesta): A privacy-preserving approach on design and analysis of digital epidemiology chronic disease tool (depict). *7th annual Primary Healthcare Partnership Forum (PriFor)*, 2015.

- [52] S. Samet and A. Miri. Privacy-preserving protocols for perceptron learning algorithm in neural networks. In *2008 4th International IEEE Conference Intelligent Systems*, volume 2, pages 10–65. IEEE, 2008.
- [53] S. Samet and A. Miri. Privacy-preserving bayesian network for horizontally partitioned data. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 3, pages 9–16. IEEE, 2009.
- [54] S. Samet and A. Miri. Privacy-preserving back-propagation and extreme learning machine algorithms. *Data & Knowledge Engineering*, 79:40–61, 2012.
- [55] S. Samet, A. Miri, and L. Orozco-Barbosa. Privacy preserving k-means clustering in multi-party environment. In *SECRYPT*, pages 381–385, 2007.
- [56] S. Senn and W. Richardson. The first t-test. *Statistics in medicine*, 13(8):785–803, 1994.
- [57] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.
- [58] J. F. Traub, Y. Yemini, and H. Woźniakowski. The statistical security of a statistical database. *ACM Transactions on Database Systems (TODS)*, 9(4):672–679, 1984.
- [59] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 639–644. ACM, 2002.

- [60] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215. ACM, 2003.
- [61] J. Vaidya and C. Clifton. Privacy preserving naïve bayes classifier for vertically partitioned data. In *SDM*, pages 522–526. SIAM, 2004.
- [62] J. Vaidya and C. Clifton. Privacy-preserving decision trees over vertically partitioned data. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 139–152. Springer, 2005.
- [63] L. Willenborg and T. De Waal. *Statistical disclosure control in practice*. Number 111. Springer Science & Business Media, 1996.
- [64] A. C. Ya

o. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 160–164. IEEE, 1982.

# Appendix A

## UML Diagrams

Use case diagrams, database scheme and the class diagram of the secure computation framework have been shown below.

<b>UseCase</b>	<b>Registration</b>
Context	A new user requests to authority server for registering into secure computation framework
Scope	Secure Computation Framework Authority Server
Actor	Researcher, Data Owner, Administrator
Preconditions	The user must be a new user and should be a researcher or a data owner
Trigger	User clicks on the register button or administrator clicks on a create new user button
Successful Sequence	<p>Scenario 1: User registraion by the user:  User must complete registration form and the system prompts for information such as name, email address, password and etc.  User enters correct information  User clicks submit button  System verifies the information and creates unactivated account  Administrator manually verifies the user and activates the account  Use case ends</p> <p>Scenario 2: User registration by the administrator  Administrator must complete registration form and the system prompts for information such as name, email address, password and etc.  Administrator enters correct information  Administrator clicks submit button  System verifies the information and creates activated account  Use case ends</p>
Failed Sequence	<p>Administrator or user cancels registration  Administrator or user does not enter correct information (e.g. incorrect email address format)  Administrator manually checks the user and disables the account</p>

Figure 15: registration use case diagram

<b>UseCase</b>	<b>Login</b>
Context	Users log in to the system using their email and password
Scope	Secure Computation Framework Authority Server
Actor	Researcher, Administrator
Preconditions	The user must be an activated registered user and should be a researcher or an administrator
Trigger	User clicks on the login button after entering their email and password
Successful Sequence	<p>The system prompts for email and password</p> <p>User enters correct information</p> <p>User clicks login button</p> <p>System verifies the entered email and password and creates required variables such as session and cookie and the user will be signed in and will be redirected to their corresponding dashboard</p> <p>Use case ends</p>
Failed Sequence	<p>User enters incorrect email and/or password and the systems describes the reason</p> <p>User is not activated and the systems describes the reason</p>

Figure 16: login use case diagram

UseCase	Key Generation
Context	Data owner establishes a Paillier cryptosystem for secure computation protocols
Scope	Data Owner Web Services
Actor	Data Owner
Preconditions	The data owner must be verified with online web service
Trigger	Authority server sends a query to the data owner
Successful Sequence	<p>Authority server sends a query and a unique id to a data owner</p> <p>Data owner stores the unique id and generates the private key and public keys</p> <p>Data owner sends the public keys to the authority server</p> <p>Use case ends</p>
Failed Sequence	<p>Query submitted by the authority server is invalid</p> <p>Data owner might not be able to connect to authority server because of various issues such as authority server is shut down</p> <p>Data owner does not receive an id</p>

Figure 17: key generation use case diagram

<b>UseCase</b>	<b>Query Submission</b>
Context	Researcher sends a preferred query to the system to get an aggregated result
Scope	Secure Computation Framework Authority Server, Data Owner Web Services
Actor	Researcher
Preconditions	The user must be an activated registered user and should be a researcher
Trigger	Researcher clicks on the submit button on the create query page after entering and selecting required parameters such as statistical analysis methods, fields and conditions.
Successful Sequence	<p>Researcher clicks on Submit Query on the left menu</p> <p>Researcher selects preferred statistical analysis method</p> <p>Researcher selects preferred column(s)</p> <p>Researcher adds as many conditions and/or condition blocks as needed and enters or selects required parameters</p> <p>User clicks submit button</p> <p>Authority server verifies the query and sends it to data owners</p> <p>First data owner establishes a Paillier cryptosystem and sends public keys to authority server</p> <p>Authority server distributes public keys through all data owners</p> <p>Data owners encrypt their own private inputs and send them to authority server</p> <p>Authority server exchanges encrypted data between data owners</p> <p>Data owners provide private output(s) and send it to authority server</p> <p>Authority server computes the aggregated result</p> <p>The result will be shown to the researcher</p> <p>Use case ends</p>
Failed Sequence	<p>Researcher does not select required fields or enters incorrect information in the condition fields</p> <p>Authority server does not verify the submitted query and the system prints out the reason</p> <p>Data owners' web services are not available and the system prints out the reason</p>

Figure 18: query submission use case diagram

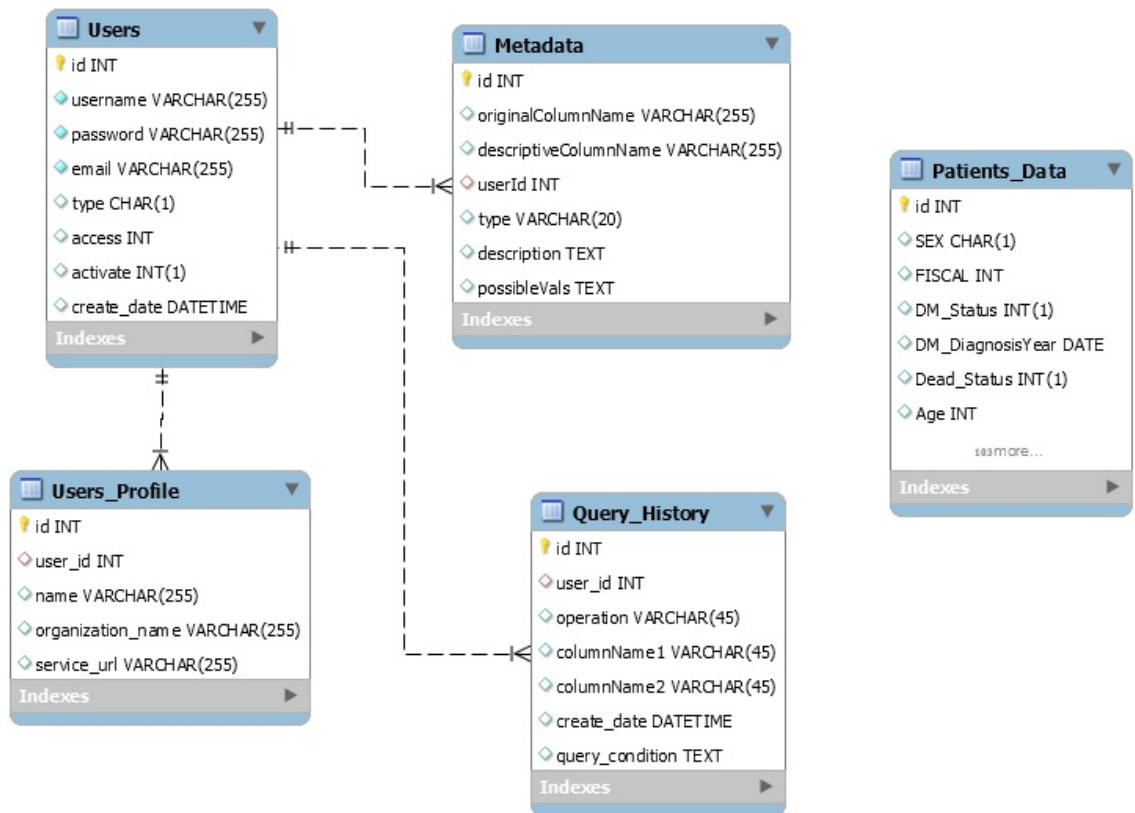


Figure 19: secure computation database scheme

