

PREDICTIVE ANALYSIS ON ENCRYPTED DATA WITH MULTI-USER ACCESS

A PROJECT REPORT

Submitted By

PRISCILLA ANDREW 312215104073

PRIYA LAKSHMI T. 312215104075

VIDYA R. 312215104125

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

SSN COLLEGE OF ENGINEERING

KALAVAKKAM 603110

ANNA UNIVERSITY :: CHENNAI - 600025

April 2019

ANNA UNIVERSITY : CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report titled “**PREDICTIVE ANALYSIS ON ENCRYPTED DATA WITH MULTI-USER ACCESS**” is the *bonafide* work of “**PRISCILLA ANDREW (312215104073), PRIYA LAKSHMI T. (312215104075), and VIDYA R. (312215104125)**” who carried out the project work under my supervision.

DR. CHITRA BABU
HEAD OF THE DEPARTMENT

Professor,
Department of CSE,
SSN College of Engineering,
Kalavakkam - 603 110

MR.V.
BALASUBRAMANIAN
SUPERVISOR
Assistant Professor,
Department of CSE,
SSN College of Engineering,
Kalavakkam - 603 110

Place:

Date:

Submitted for the examination held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENTS

We thank GOD, the almighty for giving us strength and knowledge to do this project.

We would like to thank and express a deep sense of gratitude to our guide **Mr. V. BALASUBRAMANIAN**, Assistant Professor, Department of Computer Science and Engineering, for his valuable advice and suggestions as well as his continued guidance, patience and support that helped us to shape and refine our work.

Our sincere thanks to **Dr. CHITRA BABU**, Professor and Head of the Department of Computer Science and Engineering, for her words of advice and encouragement and we would like to thank our project coordinator **Dr. S. SHEERAZUDDIN**, Associate Professor, Department of Computer Science and Engineering for his valuable suggestions throughout the project.

We also express our appreciation to **Dr. S. SALIVAHANAN**, our Principal, for all the help he has rendered during this course of study.

We would like to extend our sincere thanks to all the teaching and non-teaching staffs of our department who have contributed directly and indirectly during the course of our project work. Finally, we would like to thank our parents and friends for their patience, cooperation and moral support throughout our life.

PRISCILLA ANDREW

PRIYA LAKSHMI T.

VIDYA R.

ABSTRACT

Privacy and confidentiality of data is one of the growing concerns of cloud security. Although an organization's data is encrypted during transit and at rest, the organization's data is definitely not encrypted if it is processed in the cloud. Therefore unless the data is in the cloud for simply storage, data in the cloud will be unprotected for at least a small part of its life cycle. Homomorphic encryption was developed to allow the processing of encrypted data. This method requires the owner of the data to possess a key pair for encrypting the data and for decrypting the result from the cloud. Anyone not in possession of the key pair will not be able to access the result. Hence multi-user access of the processed result is not possible. Our system proposes a method to allow multiple users to access the results without having access to the original data or the key pair.

TABLE OF CONTENTS

LIST OF FIGURES	vii
1 INTRODUCTION	1
1.1 Motivation	1
1.2 Proposed System	2
1.3 Homomorphic Encryption	2
1.4 Proxy Re-encryption	3
2 LITERATURE SURVEY	5
2.1 Private Predictive Analysis on Encrypted Medical Data	5
2.2 Multi-User Searchable Symmetric Encryption with Dynamic Updates for Cloud Computing	6
2.3 Sharing Data Homomorphically Encrypted with Different Encryption Keys	8
2.4 Unidirectional Proxy Re-encryption	9
2.5 Data Analytics on Encrypted Data Elements	10
2.6 Encrypted Data Processing with Homomorphic Re-Encryption . .	12
2.7 Scalable and Secure Logistic Regression via Homomorphic Encryption	14
3 LOGISTIC REGRESSION	16
3.1 Preprocessing of Data	16
3.2 Logistic Regression Model	19

4	PAILLIER CRYPTOSYSTEM	21
4.1	Algorithm	21
4.1.1	Key Generation	21
4.1.2	Encryption	22
4.1.3	Decryption	22
4.2	Homomorphic Properties of Paillier Cryptosystem	23
4.2.1	Homomorphic Addition of Plain Text	23
4.2.2	Homomorphic Multiplication of Plain Text	23
4.2.3	Homomorphic Multiplication of Plain Text with a Constant	24
5	PROXY RE-ENCRYPTION	25
5.1	Proxy Re-encryption for Paillier Cryptosystem	25
5.2	Implementation	26
6	PROBLEM DEFINITION AND PROPOSED SYSTEM	29
6.1	Problem Statement	29
6.2	System Design	29
6.3	Predictive Analysis on Encrypted Data	32
6.4	Multiple User Access of Encrypted Result	33
7	IMPLEMENTATION AND RESULTS	35
7.1	Building the Logistic Regression Model	35
7.2	Paillier Encryption and Decryption	38
7.3	Prediction on Encrypted Data	39
7.4	Proxy Re-encryption between Two Users	41

7.5 Deploying the Application	44
8 CONCLUSION AND FUTURE WORK	50
REFERENCES	51

LIST OF FIGURES

2.1	Private Predictive Analysis on Encrypted Medical Data - Cloud Service	6
2.2	Architecture of Multi-User Searchable Symmetric Encryption with Dynamic Updates	7
2.3	Homomorphic Proxy Re-encryption for Image Sharing	9
2.4	Data Analytics on Encrypted Data Elements	11
2.5	Encrypted data processing with Homomorphic Re-Encryption - System Model	13
3.1	Data Count Plot	16
3.2	Preprocessing of Data	17
5.1	Proxy Re-encryption Scheme	28
6.1	System Architecture	30
6.2	Use Case Diagram	31
6.3	Sequence Diagram - Predictive Analysis on Encrypted Data	33
6.4	Sequence Diagram - Multiple User Access of Encrypted Result . .	34
7.1	Chi-Square Test	36
7.2	Cost Function	37
7.3	Confusion Matrix	38
7.4	Paillier Cryptosystem Example	39
7.5	Plain Text Analysis - Positive Outcome	40
7.6	Cipher Text Analysis - Positive Outcome	40
7.7	Plain Text Analysis - Negative Outcome	41
7.8	Cipher Text Analysis - Negative Outcome	41
7.9	Code Snippet for Computing the Difference between two Paillier Encrypted Numbers	42
7.10	Proxy Re-encryption Example	43
7.11	Flow of Data	45
7.12	Directory Structure of the Cloud Service on Heroku	45
7.13	Directory Structure of the Client Application on Django	46
7.14	Home Page	47
7.15	Account Confirmation	47

7.16	Client Data	48
7.17	Encrypted Client Data	48
7.18	Encrypted Result of Analysis	49
7.19	Decrypted Result of Analysis	49

CHAPTER 1

INTRODUCTION

Data is the key to all research and advancements in technology. Reports claim that the amount of data collected in the last two years is more than the entire previous history. This enormous size of data cannot be processed locally. A large number of organizations are outsourcing this task of processing large amounts of data to third-party service providers. This raises concerns regarding the privacy and confidentiality of sensitive information.

1.1 Motivation

Although homomorphic encryption presents a feasible solution towards preserving the privacy and confidentiality of data, it poses a few challenges. The major requirement of asymmetric encryption is the possession of a proper key pair. Anyone with access to the key pair can decrypt the cipher text. However in computation tasks such as analysis of data, the result of the analysis may need to be accessed by multiple users who are not in possession of the key pair. Therefore measures have to be taken to ensure that the result can be accessed by third parties who do not have access to the original data or the key pair.

1.2 Proposed System

In this work, we develop a cloud service that performs predictive analysis on client sensitive data without compromising his/her privacy. This is done by encrypting the client data before it is sent to the cloud for processing using the client side application. The result of the analysis is again in encrypted form which can be decrypted by the client side application to obtain the actual results. The above can be achieved with the use of a homomorphic encryption scheme.

1.3 Homomorphic Encryption

Homomorphic encryption is a form of asymmetric encryption that allows computation on cipher texts, generating an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on the plain text [9]. Therefore with homomorphic encryption, analysis tasks such as prediction can be performed on the organization's data without compromising its confidentiality.

The Paillier cryptosystem is a homomorphic cryptosystem that possesses multiplicative and additive homomorphic properties. Let E be the encryption function and x and y be two random plain texts, then by definition,

$$E(x + y) = E(x) + E(y)$$

$$E(x * y) = E(x) * E(y)$$

Logistic regression is used for performing predictive analysis which involves fitting all the features of the regression model into a linear equation and then finding the probability of the outcome of the case under study. The usage of a linear equation makes the Paillier cryptosystem a suitable candidate for encrypting the client's data.

However, the use of a homomorphic encryption scheme poses a challenge that the result of the analysis can be accessed only by the owner of the data. There are certain scenarios, where the client may wish to share the result of the analysis with selected third parties.

1.4 Proxy Re-encryption

An asymmetric encryption scheme requires a key pair to decrypt the data. Therefore, an obvious solution to the challenge of multiple user access would be to employ a key sharing mechanism between the client and authorized third parties. But a key sharing mechanism would require updating the keys whenever a person is de-authorized, which would subsequently require re-encrypting the result, and thereby add a significant overhead. Additionally a key sharing mechanism would also require the key to be transmitted in a secure manner.

An alternate solution to the challenge of multiple user access of encrypted data is a proxy re-encryption scheme. The proxy re-encryption scheme will allow authorized third parties to recover the result of the analysis from the encrypted result without needing to be in possession of the key pair of the client. Also, the proxy re-encryption scheme does not add significant overhead as it involves only

one additional level of encryption. Let PE be the proxy re-encryption function and c_A be the cipher text obtained by encrypting a random plain text m using the public key of user A, PK_A . Then,

$$PE(c_A, PK_B) = c_B$$

where c_B is the cipher text obtained by encrypting the same plain text m using the public key of user B, PK_B .

Therefore we have enabled multiple user access in our cloud service by using the proxy re-encryption scheme illustrated in the literature [4]. Once the cloud finishes processing on the encrypted data and produces the encrypted result, the client can authorize selected third parties to access this encrypted result. These third parties can then obtain the results by using the existing proxy re-encryption scheme.

CHAPTER 2

LITERATURE SURVEY

In this chapter, we describe the techniques related to homomorphic encryption and proxy re-encryption in analysis that have been used earlier.

2.1 Private Predictive Analysis on Encrypted Medical Data

- **Idea:**

This work presented the possible application scenarios for homomorphic encryption in order to ensure privacy of sensitive medical data and described how to privately conduct predictive analysis tasks on encrypted data using homomorphic encryption [3].

- **Methodology:**

Predictive Analysis: Logistic Regression and Cox Proportional Hazard Model

Encryption: Homomorphic encryption scheme

Verification: This is required for any outsourced computation. A basic solution is as follows - simply outsource computation to multiple servers and check locally that the outputs agree once the computations are returned and decrypted.

- **Results:**

The final result of this work involved a working implementation of a

prediction service running in the cloud. The cloud service makes predictions while handling only encrypted data, learning nothing about the submitted confidential medical data. Figure 2.1 [3] illustrates the flow of data in the cloud service.

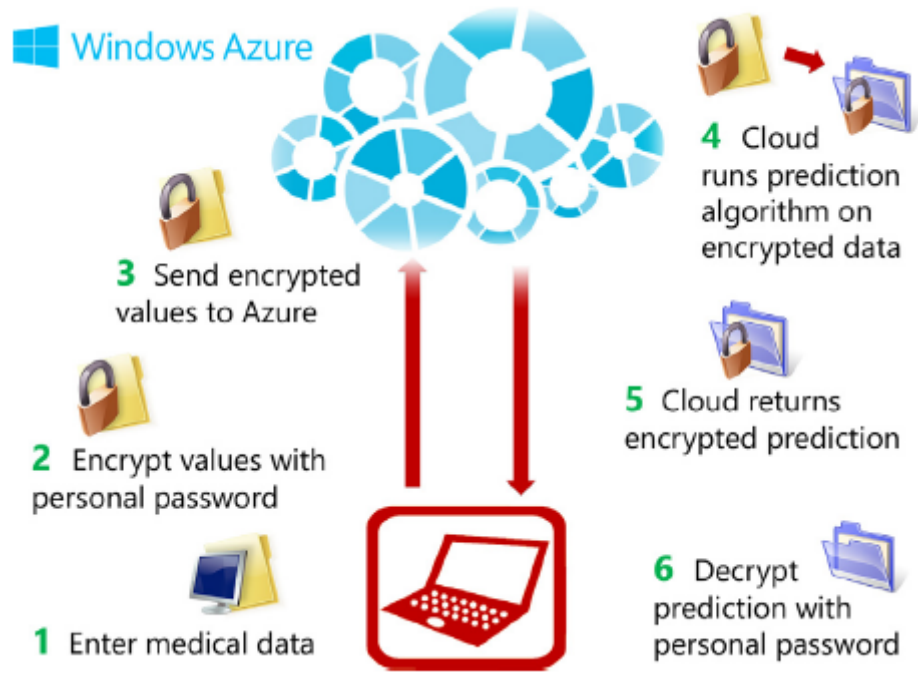


FIGURE 2.1: Private Predictive Analysis on Encrypted Medical Data - Cloud Service

2.2 Multi-User Searchable Symmetric Encryption with Dynamic Updates for Cloud Computing

- **Idea:**

To implement a searchable symmetric scheme that allows secure and efficient search or updation of data by multiple users [1]. This scheme is applicable to the usage scenario where the data owner encrypts sensitive

files and shares them among multiple users, and it allows secure and efficient searches/updates.

- **Methodology:**

This scheme used key distribution and re-encryption to achieve multi-user access. The usage of key distribution and re-encryption avoids the issues caused by key sharing. The scheme is based on an index structure where a bit matrix is combined with two static hash tables, pseudo random functions and hash functions. Figure 2.2 [1] illustrates the system architecture of this scheme.

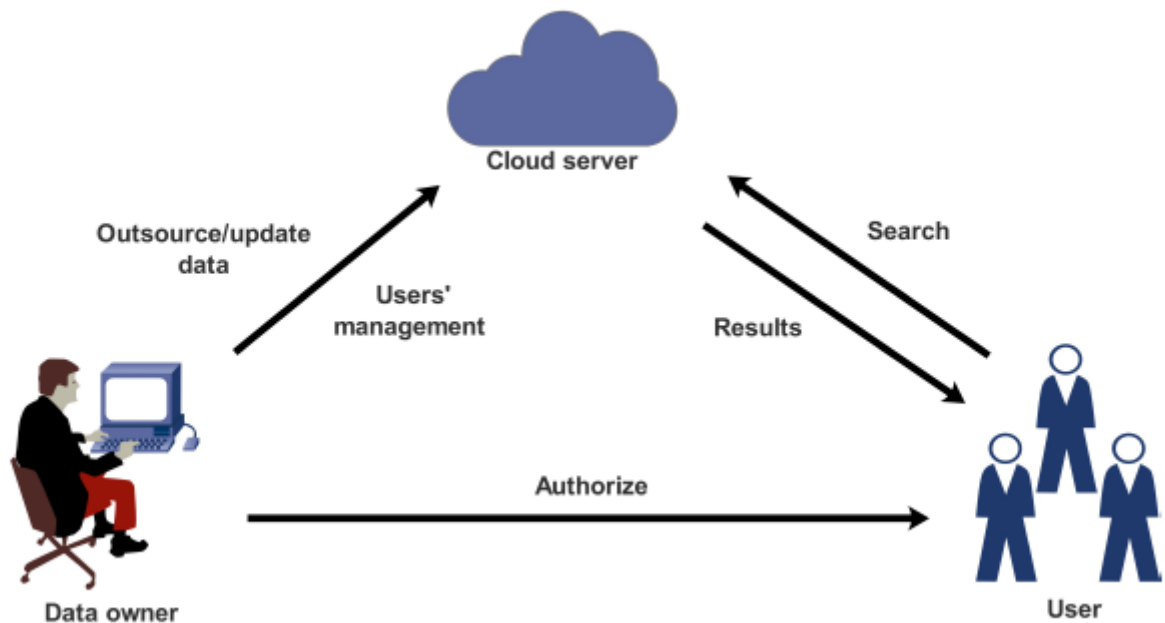


FIGURE 2.2: Architecture of Multi-User Searchable Symmetric Encryption with Dynamic Updates

- **Results:**

Using this scheme, each user can perform search operations via his/her unique keys and other are prevented from searching on behalf of a user.

2.3 Sharing Data Homomorphically Encrypted with Different Encryption Keys

- **Idea:**

This work proposed a homomorphic based proxy re-encryption (HPRE) solution that allows different users to share data they outsourced homomorphically encrypted using their respective public keys with the possibility to process such data remotely [4].

- **Methodology:**

The proxy re-encryption solution was based on the idea of sharing a secret seed between the users and subtraction of two encrypted numbers.

Paillier Cryptosystem: This work proposed a new way to compute the difference between Paillier encrypted data.

Secure Linear Congruential Generator: The scheme requires the cloud to securely generate a pseudo random sequence, that is, a Paillier encrypted random sequence of integers. The proposed generator is LCG (Linear Congruential Generator).

- **Results:**

The homomorphic based proxy re-encryption scheme proposed in this work was implemented in the case of the sharing of uncompressed images stored in the cloud. Figure 2.3 [4] illustrates the above scheme.

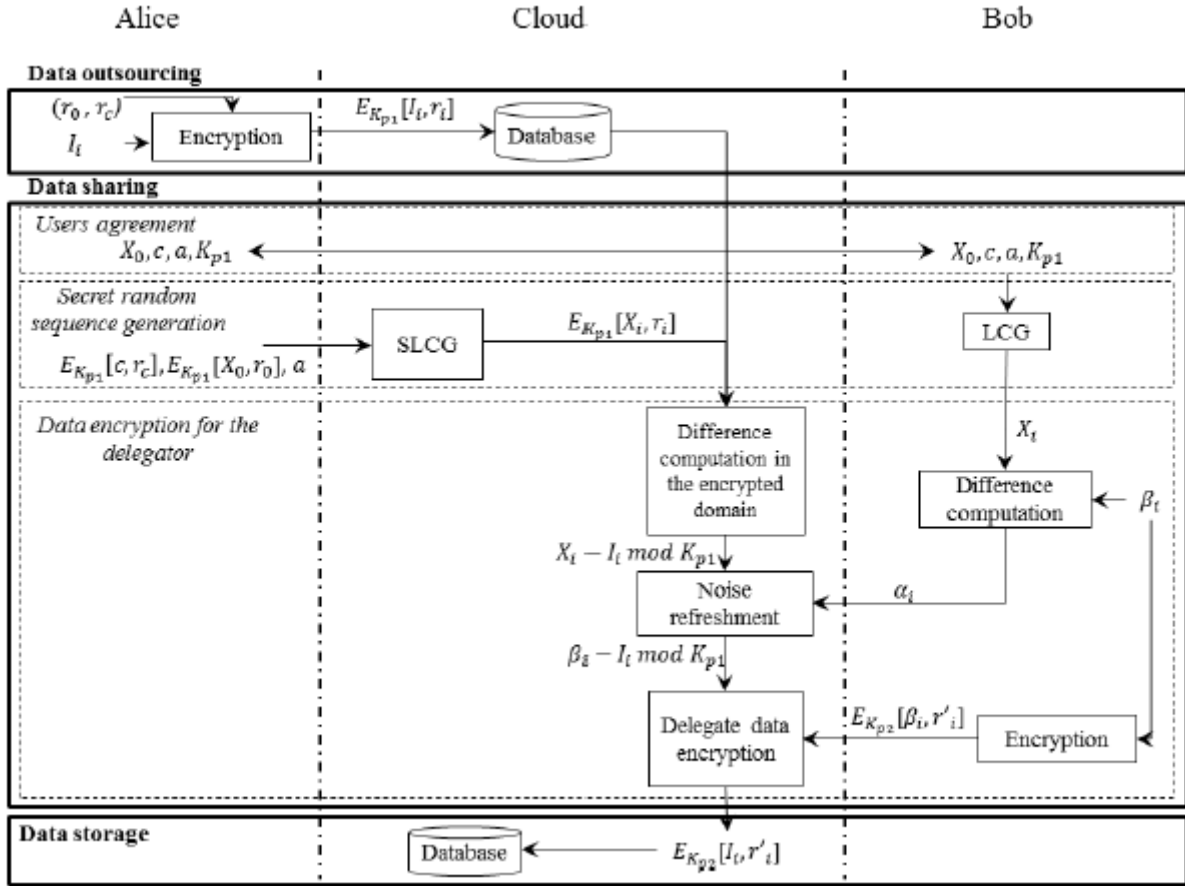


FIGURE 2.3: Homomorphic Proxy Re-encryption for Image Sharing

2.4 Unidirectional Proxy Re-encryption

- **Idea:**

This work proposed a scheme for performing unidirectional proxy re-encryption by generating a separate re-encryption key [5].

- **Methodology:**

The different methods proposed in this work are:

1. Generating a first key pair comprising a public key (PK) and a private key (SK) and then generating a re-encryption key that changes

encryption under a first public key PK_a , into encryption under a second public key PK_b as $RK_{a \rightarrow b}$.

2. Encrypting a message m under a public key PK_a , producing a cipher text c_a , re-encrypting the cipher text c_a using the re-encryption key $RK_{a \rightarrow b}$ that changes cipher texts under PK_a into cipher texts under PK_b to recover a message m .
3. Encrypting a message m under a public key PK producing a first-level cipher text c that cannot be re-encrypted and decrypting the first-level cipher text c using the private key PK .

- **Results:**

This work can be programmed strictly as a software program, or developed as hardware.

2.5 Data Analytics on Encrypted Data Elements

- **Idea:**

Secure protocols are generally employed to allow computation involving data from multiple parties to be performed without the data being revealed to any of the involved or third parties. The result can however be made available to the parties.

The data analytics system maps the encrypted data elements to an analytics space, performs data analytics based on the mapped data elements, and distributes, via a computing device, results of the data analytics to an information retrieval system [2].

- **Methodology:**

Figure 2.4 [2] illustrates the steps involved in this work.

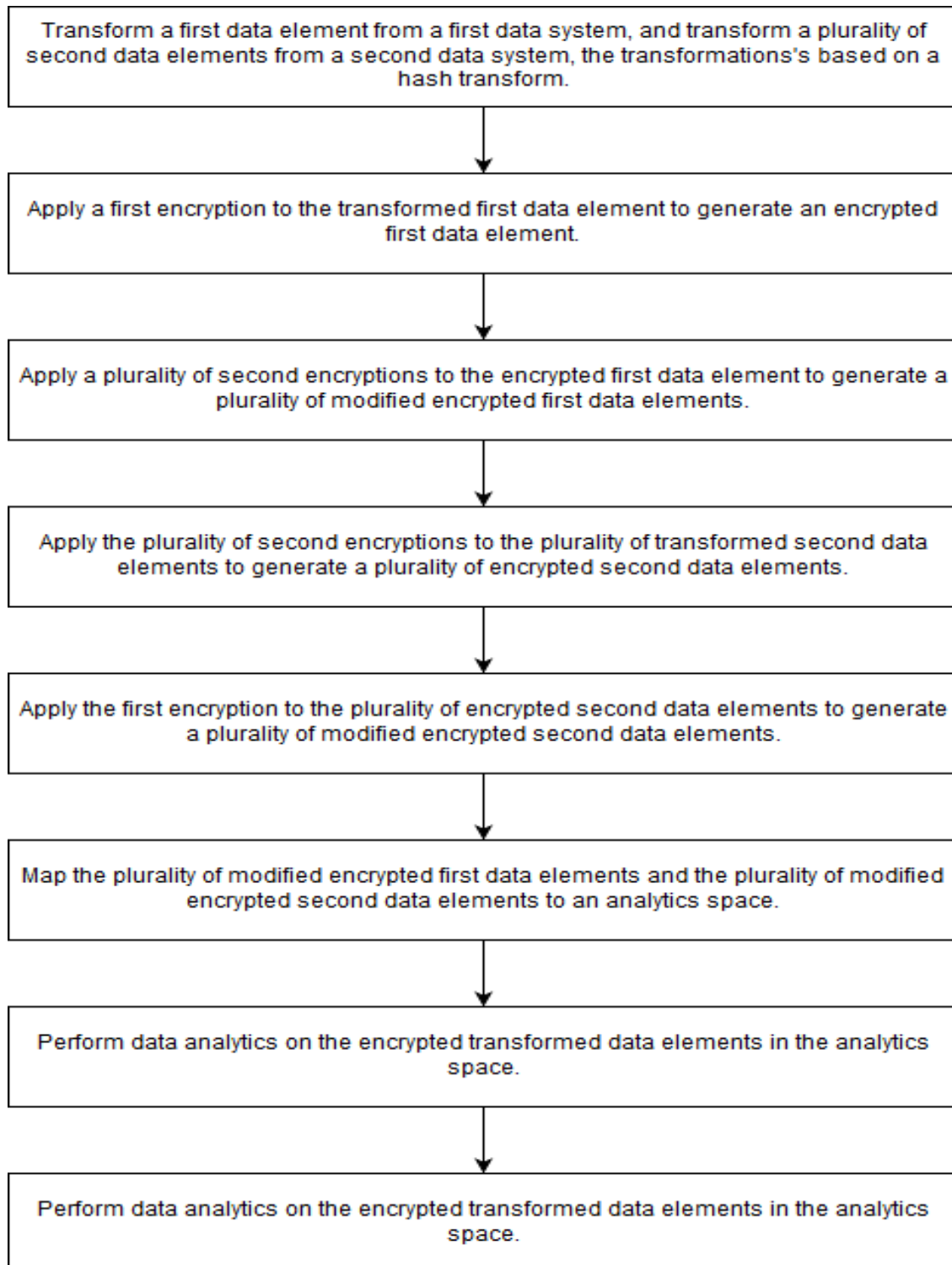


FIGURE 2.4: Data Analytics on Encrypted Data Elements

- **Results:**

In most existing protocols, privacy and data analytics are intertwined, thereby making arbitrary data analytic tasks infeasible. This method successfully decouples privacy protection and data analytics.

2.6 Encrypted Data Processing with Homomorphic Re-Encryption

- **Idea:**

Cloud computing relieves users from the burden of managing large information systems and breaks the bottleneck of local resources. Privacy of data is ensured by encryption. However encryption comes with its own challenges [6].

First, encryption restricts processing of cipher texts. This can be overcome with the use of a homomorphic encryption (HE) scheme. Second, the operations on cipher texts is limited. A fully homomorphic encryption scheme (FHE) allows arbitrary operations on cipher text, but this adds computation overhead. Third, a large computation overhead is involved especially when processing large amounts of input data.

This work proposes a privacy-preserving data processing (PPDP) system to overcome the above three problems.

- **Methodology:** The privacy-preserving data processing (PPDP) system involves four entities:

1. Data service provider (DSP)

2. Access control server (ACS)
3. Data provider (DP)
4. Data requesters (DR)

The DSP collects and stores the encrypted data outsourced by the DPs. After getting the request from the DR, the DSP cooperates with the ACS to process the stored data according to DR demands. Finally, the cipher text processing results are returned to the eligible DR that can obtain the plain processing results. Figure 2.5 [6] illustrates the system model.

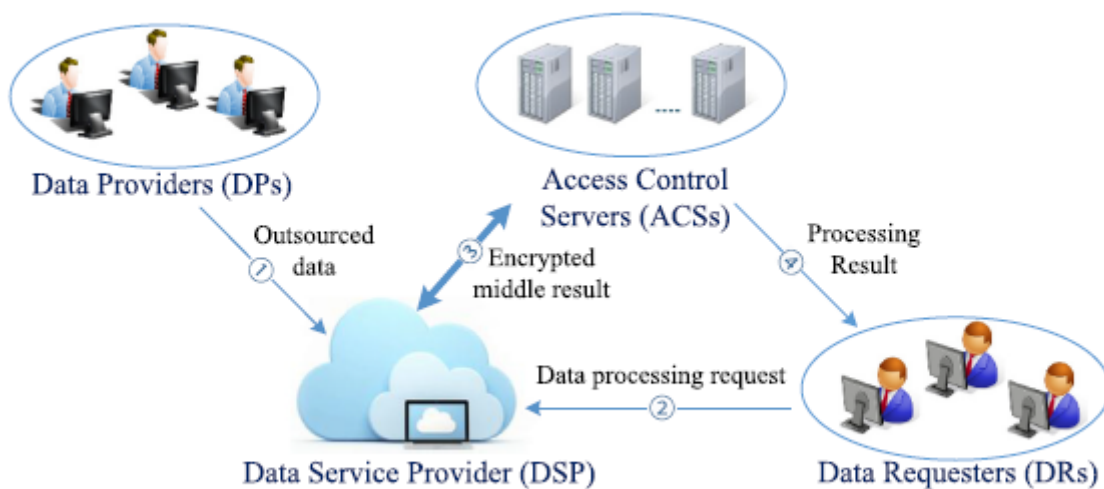


Fig. 1. System Model.

FIGURE 2.5: Encrypted data processing with Homomorphic Re-Encryption - System Model

• Results:

With the help of of a Data Service Provider (DSP) and an Access Control Server (ACS), the PPDP system can support seven basic operations over cipher texts, which include addition, subtraction, multiplication, sign acquisition, comparison, equivalent test , and variance .

2.7 Scalable and Secure Logistic Regression via Homomorphic Encryption

- **Idea:**

This work proposes a method for securing both the training and testing data in logistic regression via homomorphic encryption. Despite the non-polynomial tasks of training and predicting in logistic regression, the paper shows that only additively homomorphic encryption is needed to build the system [7].

- **Methodology :**

In this work, both the training data and the testing data are protected under encryption, so data secrecy is ensured. In addition, the output of the system can achieve differential privacy. The proposed system is constructed via the following steps [7]:

1. The original logistic regression is converted into a homomorphism-aware logistic regression via functional approximations.
2. Additive homomorphic encryption is used with the homomorphism-aware logistic regression. Differential privacy is also added to the system.
3. This work demonstrates the instantiation of the proposed system with Paillier, learning with errors based (LWE), and ring-LWE-based encryption.

- **Results:**

This work produces a system that is both secure and scalable being able to handle very large datasets of size up to hundreds of millions of records (10^8).

CHAPTER 3

LOGISTIC REGRESSION

The main functionality of the proposed system is predictive analysis. The analysis is performed using logistic regression as the the target variable has only two possible outcomes - '0' and '1'.

3.1 Preprocessing of Data

In the proposed system, we perform predictive analysis on the Framingham heart study data set [8]. The data set contains 4240 records. Figure 3.1 represents graphically the number of positive and negative records. The purpose of the analysis is to predict if a person has a ten year risk of a cardiovascular heart disease (CHD).

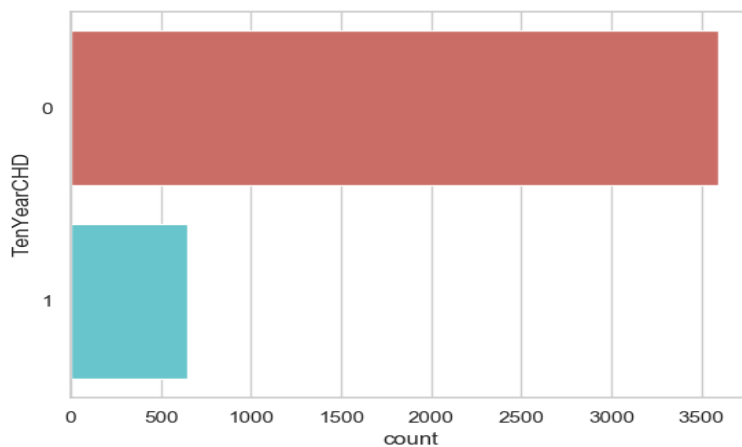


FIGURE 3.1: Data Count Plot

The data set consists of fifteen independent features - male, age, education, current smoker, cigarettes per day, BP medicines, prevalent stroke, prevalent

hypertension, diabetes, total cholesterol, systolic pressure, diabolic pressure, BMI, heart rate and glucose. The target feature is binary where 1 indicates ‘10 year risk of cardiovascular heart disease’ and 0 indicates ‘No 10 year risk of cardiovascular heart disease’.

Figure 3.2 illustrates the steps involved in preparing the data for analysis.

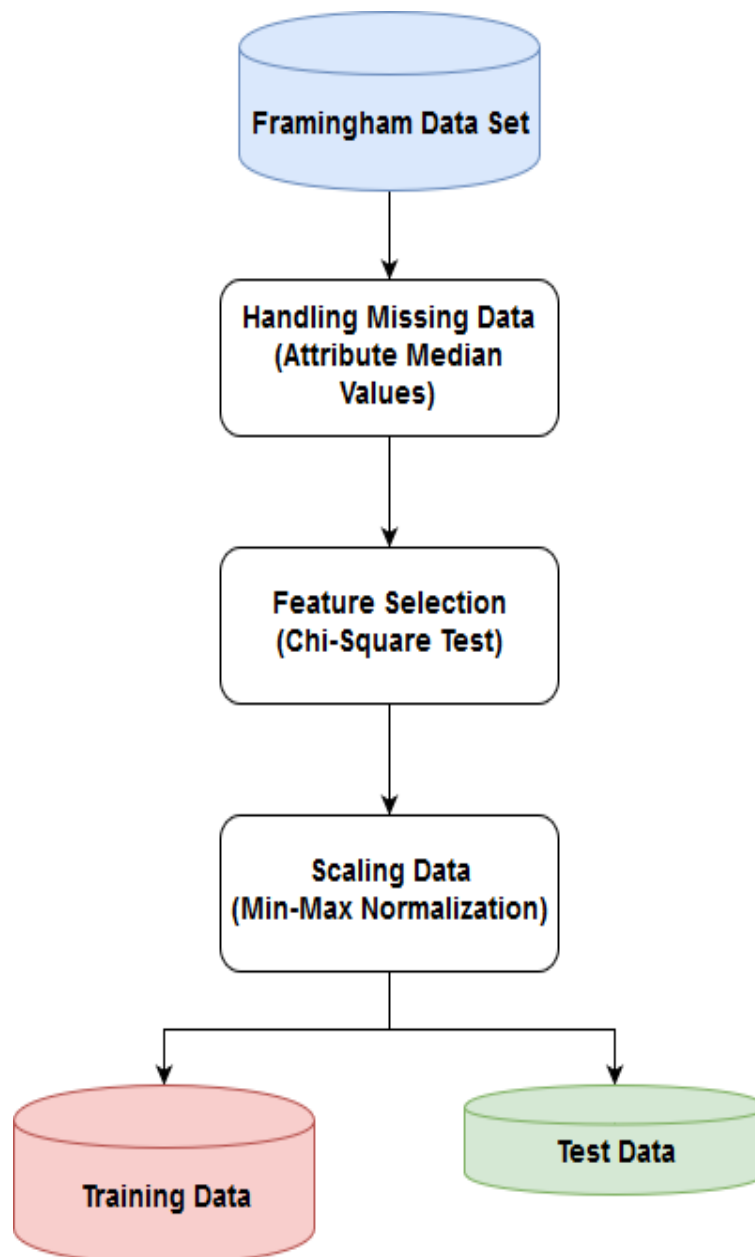


FIGURE 3.2: Preprocessing of Data

The missing values in the data set were computed using the median value of the corresponding features.

Feature selection was performed using the Chi-Square test. The Chi-Square test is a procedure for testing if two categorical features are related in some population. The null hypothesis states that there is no association between the two features. Features that satisfy this hypothesis are excluded from the list of features required for logistic regression.

The Chi-Square test was performed between each independent feature and the target feature which led to the elimination of the features - current smoker, cigarettes per day and heart rate.

Finally the data set was scaled to the range $[-1,1]$ using the min-max normalization formula

$$v' = \frac{(v - \min)}{(\max - \min)} * (\text{newmax} - \text{newmin}) + \text{newmin}$$

where,

v is the feature under consideration,

v' is the normalized value of the feature v ,

\min is the minimum value of v that has been observed in the data set,

\max is the maximum value of v that has been observed in the data set,

$\text{newmin} = -1$ and

$\text{newmax} = 1$.

3.2 Logistic Regression Model

A logistic regression model uses a logistic function to model a binary dependent variable. Logistic regression is used to estimate the parameters of a logistic regression model [10]. The logistic regression model is then used to predict the likelihood of the occurrence of the event under study.

Logistic regression is suitable for predictive analysis when the dependent variable is dichotomous, that is, when the dependent variable has only two possible outcomes.

The output of the logistic regression model is the probability of 10 year risk of CHD. The threshold is set to 0.5 such that if $P(10 \text{ Year CHD}) < 0.5$ then the result is '0' else the result is '1'.

Gradient descent has been used for training the logistic regression model. It is an optimization algorithm, that is based on a convex function, that modifies its parameters iteratively to minimize a given function to its local minimum. It measures the change in all the weights of the model with regard to the change in error.

In medical diagnosis, test sensitivity is the ability of a test to correctly identify those with the disease (true positive rate), whereas test specificity is the ability of the test to correctly identify those without the disease (true negative rate).

Let TP be the number of true positives, TN the number of true negatives, FP the number of false positives and FN the number of false negatives produced by a model. Then,

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

A type I error corresponds to the false positive predictions while a type II error corresponds to the false negative predictions.

The trained model was found to be highly specific than sensitive based on the number of type I and type II errors.

CHAPTER 4

PAILLIER CRYPTOSYSTEM

The Paillier cryptosystem is an asymmetric algorithm for public key cryptography. The scheme is an additive homomorphic cryptosystem [11]. The Paillier cryptosystem also has a property that a particular plain text may have many different cipher texts. This is due to the inclusion of randomness into the encryption. This feature makes the Paillier cryptosystem resistant to various cryptographic attacks.

4.1 Algorithm

In Paillier cryptosystem, given only the public-key and the encryption of m_1 and m_2 , one can compute the encryption of $m_1 + m_2$.

4.1.1 Key Generation

Two random prime numbers p and q of equal length are chosen. Equal length of the prime numbers ensures the condition $\gcd(pq, (p-1)(q-1)) = 1$. The key generation is as follows:

$$K_p = p * q$$

$$K_s = \phi(K_p)$$

where ϕ is the Euler Totient function and $\phi(K_p) = (p-1) * (q-1)$

$$g = 1 + K_p$$

$$\mu = \phi(K_p)^{-1} = K_s^{-1}$$

The public key is (K_p, g) and the private key is (K_s, μ) .

4.1.2 Encryption

Given a public key K_p , the message m can be encrypted as follows:

$$c = E(m, r) = (1 + mK_p)r^{K_p} \bmod K_p$$

where r is a random integer, $0 < r < K_p$, or

$$c = E(m, r) = g^m r^{K_p} \bmod K_p$$

where r is a random integer, $0 < r < K_p$ and $g = 1 + K_p$

4.1.3 Decryption

Given a public key K_p and private key K_s , the cipher text c can be decrypted as follows:

$$m = \frac{(c^{K_s} - 1)K_s^{-1} \bmod K_p^2}{K_p} \bmod K_p$$

To accomodate negative integers into the Paillier cryptosystem, an additional step has been added to the decryption. The message m obtained from the previous step is modified as follows :

$$m' = ((m + \lfloor \frac{K_p}{2} \rfloor) \bmod K_p) - \lfloor \frac{K_p}{2} \rfloor$$

4.2 Homomorphic Properties of Paillier Cryptosystem

The Paillier cryptosystem has the following homomorphic properties.

4.2.1 Homomorphic Addition of Plain Text

The product of two cipher texts will decrypt to the sum of their corresponding plain texts.

$$D(E(m_1, r_1) \cdot E(m_2, r_2) \bmod K_p^2) = m_1 + m_2 \bmod K_p$$

4.2.2 Homomorphic Multiplication of Plain Text

An encrypted plain text raised to the power of another plain text will decrypt to the product of the two plain texts.

$$D(E(m_1, r_1)^{m_2} \bmod K_p^2) = m_1 \cdot m_2 \bmod K_p$$

$$D(E(m_2, r_2)^{m_1} \bmod K_p^2) = m_1 \cdot m_2 \bmod K_p$$

4.2.3 Homomorphic Multiplication of Plain Text with a Constant

An encrypted plain text raised to a constant k will decrypt to the product of the plain text and the constant.

$$D(E(m_1, r_1)^k \bmod K_p^2) = k.m_1 \bmod K_p$$

CHAPTER 5

PROXY RE-ENCRYPTION

Proxy re-encryption (PRE) schemes are cryptosystems which allow third parties (proxies) to alter a cipher text which has been encrypted for one party, so that it may be decrypted by another [12].

5.1 Proxy Re-encryption for Paillier Cryptosystem

The proxy re-encryption scheme for Paillier cryptosystem that has been used in the proposed system is based on the literature [4].

Proxy re-encryption of Paillier encrypted data requires the calculation of the difference between two Paillier encrypted data. Computing the difference between two Paillier encrypted data introduces a constraint that the two cipher texts have to be encrypted using the same random integer r . The two users who exchange the data agree on a shared integer X .

The encrypted difference function for two plain text m_1 and m_2 is given by

$$m_1 - m_2 = \frac{E(m_1, r)E(m_2, r)^{-1} - 1 \bmod K_p^2}{K_p} \bmod K_p$$

The proxy re-encryption scheme does not add any significant overhead as it requires only one extra level of encryption and does not require the generation of any additional keys.

5.2 Implementation

Let E be the encryption function, D be the decryption function and m the plain text that is to be shared between two users.

$$c = E(m, r, PK) = (1 + mPK)r^{PK} \bmod PK$$

$$m = D(c, SK) = \frac{(c^{SK} - 1)SK^{-1} \bmod PK^2}{PK} \bmod PK$$

where, PK and SK are the public and private keys respectively.

The steps involved in the proxy re-encryption scheme are as follows :

1. Let user A be the owner of the encrypted data in the cloud and user B be the party that wishes to access this encrypted data. The plain text m is encrypted using the public key of A, PK_A and a random integer r as

$$c_A = E(m, r, PK_A)$$

2. When user A authorizes user B, a secret integer X is established between them. The integer X must be chosen such that it is much larger than any possible value of the plain text. It is known only to user A and user B.
3. The integer X is encrypted using the public key of user A and the same random integer r that was used to encrypt the original data in the cloud. The encrypted integer X is given by

$$cx_A = E(X, r, PK_A)$$

4. The difference between the encrypted integer X and the encrypted cloud data is computed using the aforementioned formula.

$$X - m = \frac{E(X, r, PK_A)E(m, r, PK_A)^{-1} - 1 \bmod PK_A^2}{PK_A} \bmod PK_A$$

5. Knowledge of the computed difference will not reveal any information about the original data or the secret integer X .
6. This difference is then encrypted using the public key of user B, PK_B and a random integer s as

$$cd_B = E(X - m, s, PK_B)$$

7. User B can now access the original data by simply decrypting the data from the previous step using his private key, SK_B and then subtracting it from the secret integer X .

$$X - m = D(cd_B, SK_B)$$

$$m = X - (X - m)$$

The above steps show that there is no intermediate decryption required to make the data accessible by user B.

Figure 5.1 illustrates the proxy re-encryption scheme for Paillier cryptosystem.

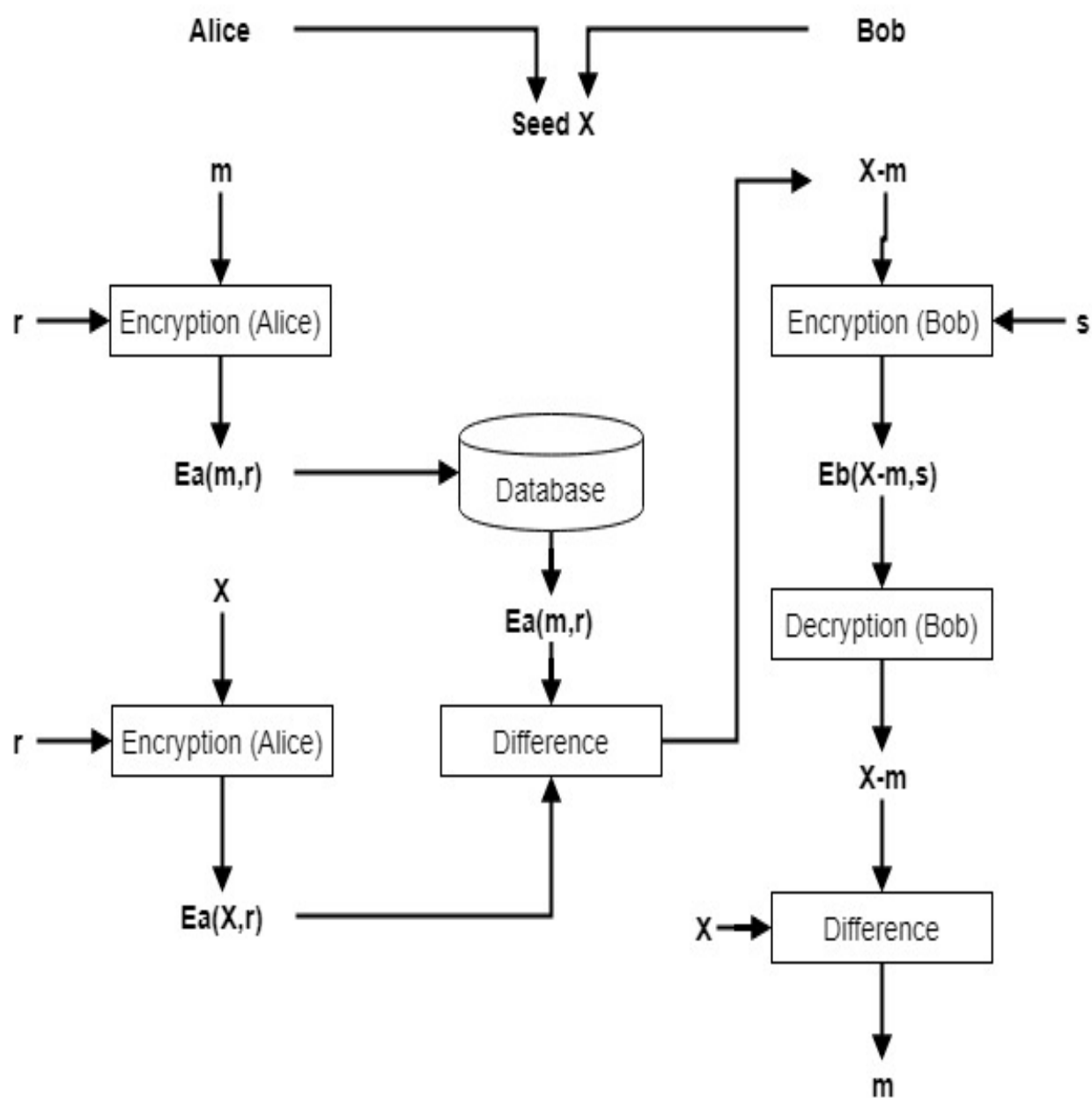


FIGURE 5.1: Proxy Re-encryption Scheme

CHAPTER 6

PROBLEM DEFINITION AND PROPOSED SYSTEM

This chapter provides a detailed description of the functionality of the modules that have been incorporated into the proposed system.

6.1 Problem Statement

The predictive analysis service consists of a client application and a cloud service. The client encrypts his/her data and sends it to the cloud, which performs predictive analysis on the encrypted data and sends the results to the client in encrypted form. The client can then recover the results using his/her decryption key. But the result cannot be accessed by multiple users who do not have the key. A key sharing mechanism will create issues related to the secure transmission of the keys. Hence re-encryption schemes can be employed to enable access to multiple users.

6.2 System Design

The goal of our system is as follows:

- To perform predictive analysis on the client's data in encrypted form.

- To allow multiple users to access the encrypted result of the analysis without key sharing.

Figure 6.1 depicts the system architecture and figure 6.2 illustrates a use case of the proposed system.

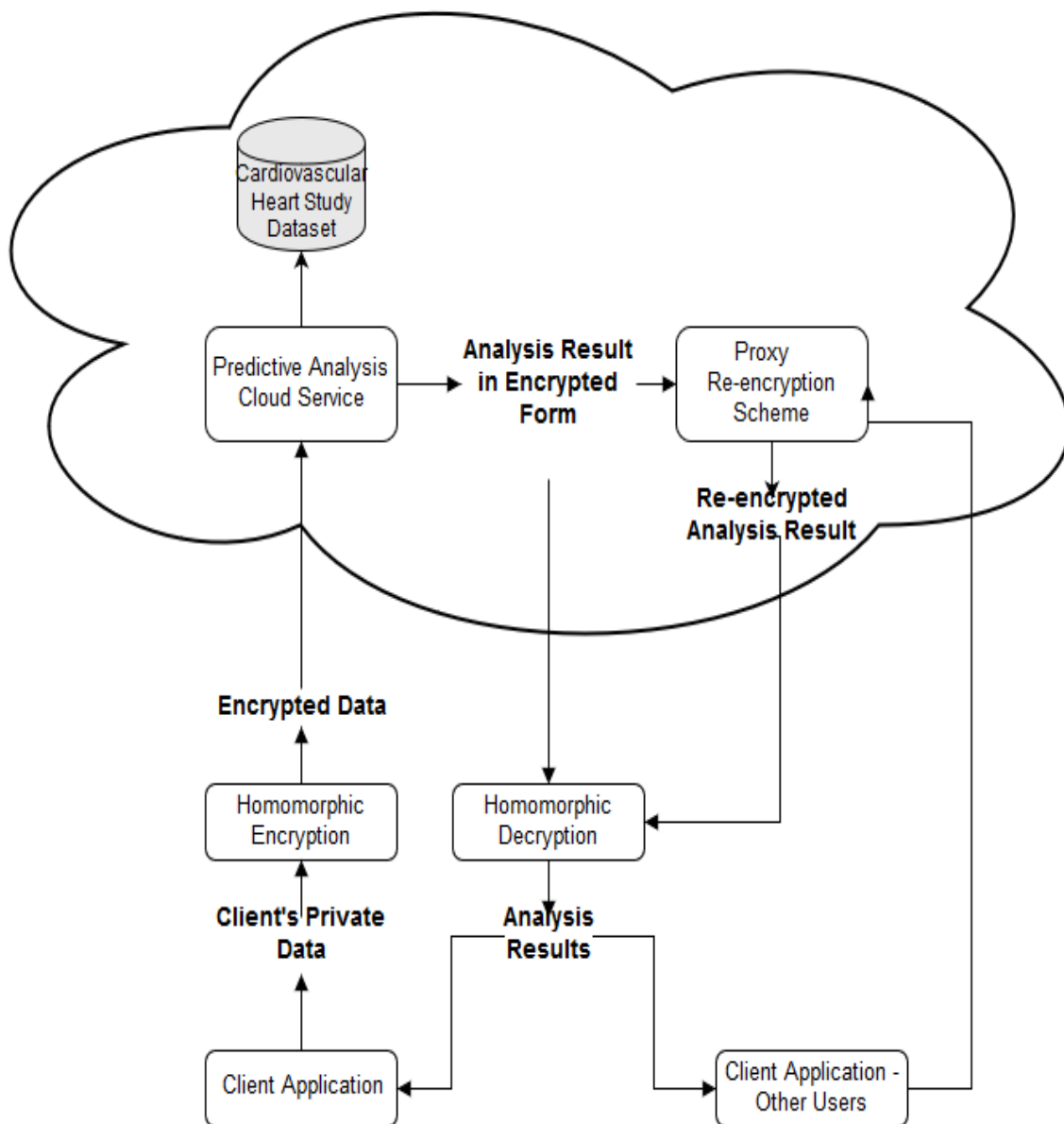


FIGURE 6.1: System Architecture

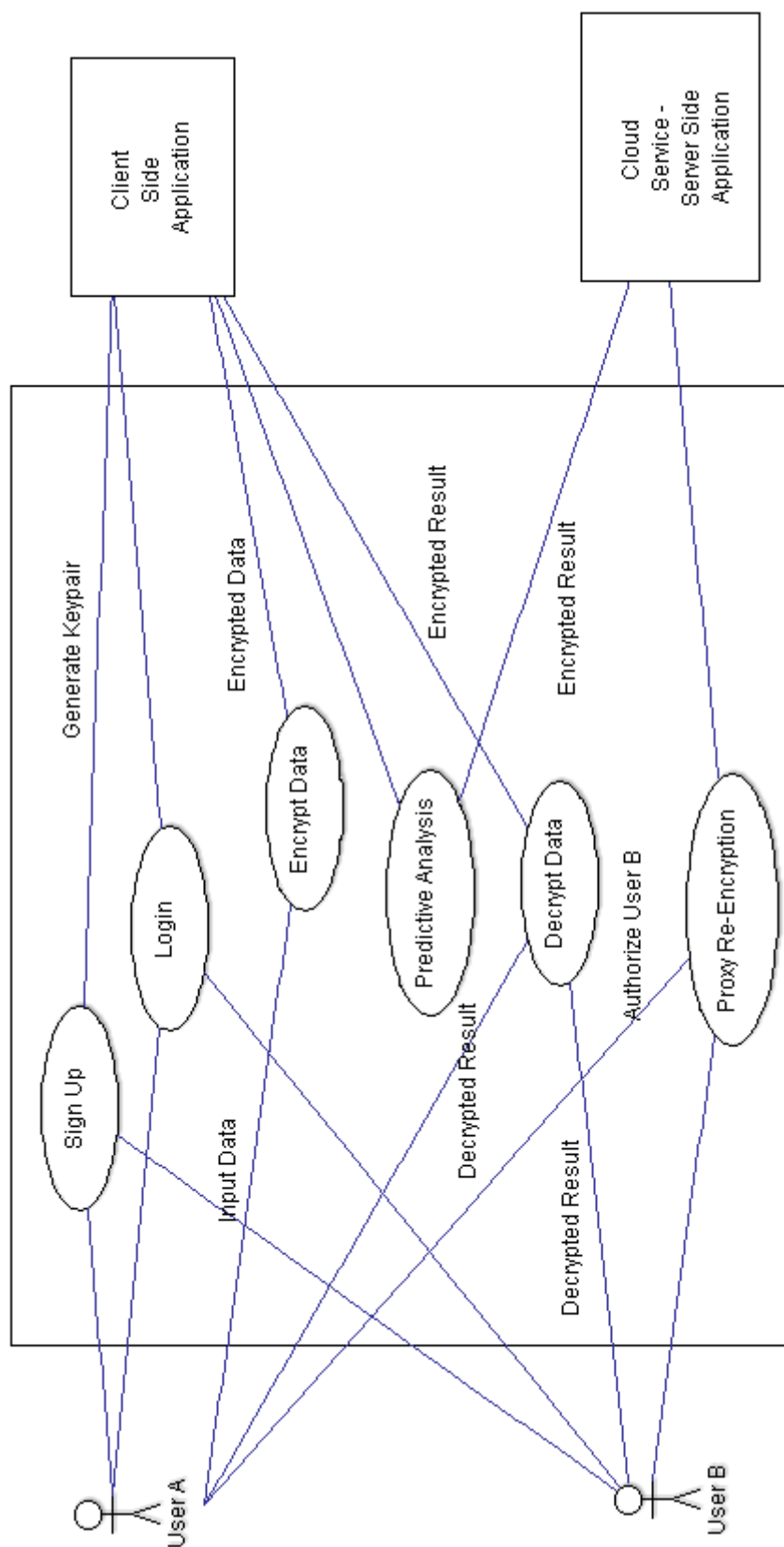


FIGURE 6.2: Use Case Diagram

6.3 Predictive Analysis on Encrypted Data

The system performs predictive analysis using logistic regression. The predictive model is initially built using historic data that is available in the cloud. The predictive model is then deployed as a service in the cloud which can be accessed by clients via the client side application.

A homomorphic cryptosystem enables the outsourcing of the predictive analysis task while maintaining the privacy of the data. The steps involved in using the predictive analysis cloud service are as follows :

1. The client enters the data that is required for the predictive analysis service using the client side application (This data is considered to be private).
2. The client's private data is encrypted using a homomorphic encryption scheme before it is sent to the cloud.
3. The predictive analysis service processes this encrypted data and produces the result which is also in encrypted form.
4. The encrypted result is then sent back to the client application.
5. The encrypted result is decrypted using the homomorphic decryption scheme to recover the result in plain text format.

The above steps are demonstrated in the sequence diagram of figure 6.3

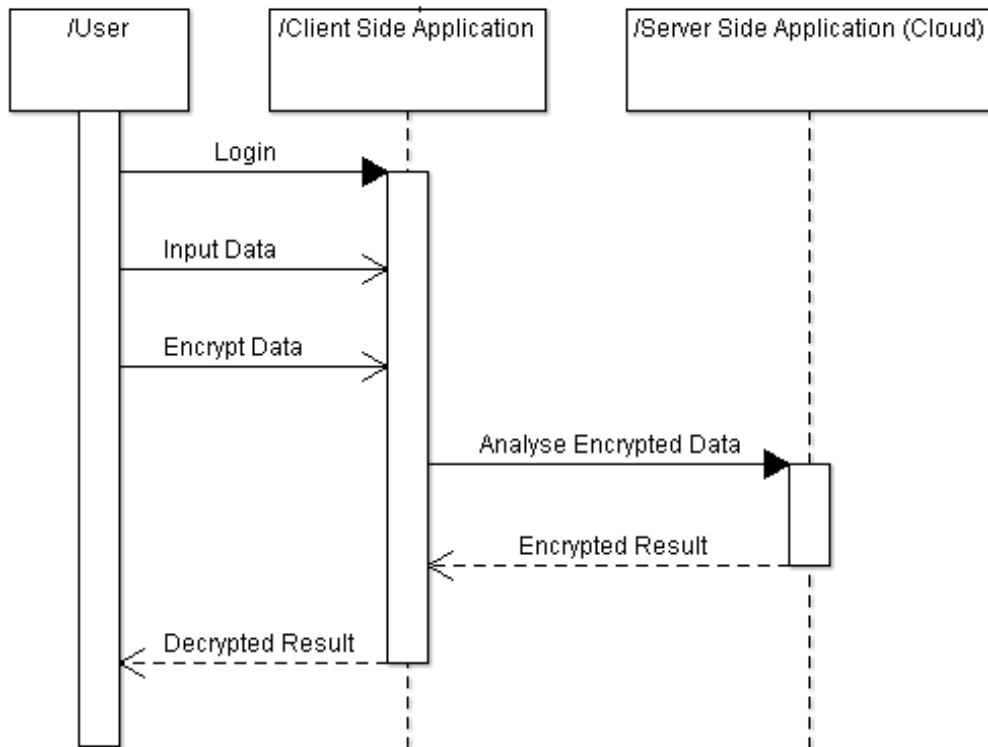


FIGURE 6.3: Sequence Diagram - Predictive Analysis on Encrypted Data

6.4 Multiple User Access of Encrypted Result

The client can authorize certain users to access the results of the predictive analysis directly from the cloud. Since the result is in encrypted form, this would require the other user to be in possession of the key that was initially used to encrypt the data. The proxy re-encryption scheme is used to overcome this disadvantage of multi-user access.

Let us consider the scenario where user B wants to access the results of analysis performed on the encrypted data of user A. The steps involved are as follows :

1. The result available in the cloud is in the form of data encrypted using the public key of user A.

2. The user B requests the cloud service for the result of user A.
3. If user B is authorized to access the result, the proxy re-encryption scheme is applied to the encrypted result of user A.
4. The encrypted result is thus transformed into the form of data encrypted using the public key of user B.
5. The user B can access this modified result and decrypt it using his/her private key to recover the result in plain text format.

The above steps are demonstrated in the sequence diagram of figure 6.4

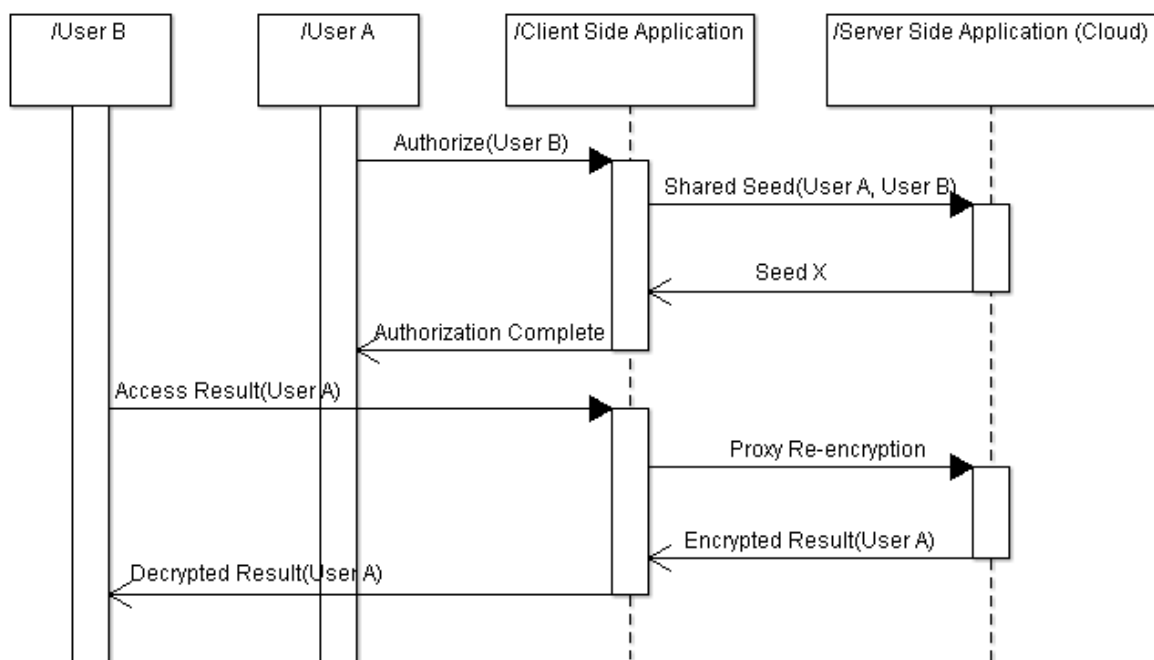


FIGURE 6.4: Sequence Diagram - Multiple User Access of Encrypted Result

CHAPTER 7

IMPLEMENTATION AND RESULTS

This chapter describes the implementation of the techniques of the proposed system, the development of the client and server side application and the results obtained.

7.1 Building the Logistic Regression Model

The data set consists of fifteen independent features and a target feature - risk of 10 year CHD. The Chi-Square test is a procedure for testing if two categorical features are related in some population. The null hypothesis states that there is no association between the two features. Features that satisfy this hypothesis are excluded from the list of features required for logistic regression.

The Chi-Square test was performed between each independent feature - male, age, education, current smoker, cigarettes per day, BP medicines, prevalent stroke, prevalent hypertension, diabetes, total cholesterol, systolic pressure, diabolic pressure, BMI, heart rate and glucose, and the target variable which led to the elimination of the features - current smoker, cigarettes per day and heart rate.

Removal of the features did not produce a significant improvement in the performance of the model. But it provided an advantage during the later stages of the system, that is, in encryption, as it reduced the number of data elements that have to be encrypted before being sent to the cloud service for analysis.

FEATURE	NULL HYPOTHESIS
male	Rejected
age	Rejected
education	Rejected
currentSmoker	*Accepted*
cigsPerDay	*Accepted*
BPMeds	Rejected
prevalentStroke	Rejected
prevalentHyp	Rejected
diabetes	Rejected
totChol	Rejected
sysBP	Rejected
diaBP	Rejected
BMI	Rejected
heartRate	*Accepted*
glucose	Rejected

FIGURE 7.1: Chi-Square Test

The resulting data set is subjected to min-max normalization and then trained using the logistic regression algorithm.

$$v' = \frac{(v - \min)}{(\max - \min)} * (\text{newmax} - \text{newmin}) + \text{newmin}$$

The processed data set was then split into two sets, that is the training data set (65%) and the test data set (35%). The Python library sklearn provides a built-in function `train_test_split`, which randomly splits the data based on the required percentage.

Gradient descent is used while training the model. The goal is to minimize the cost function to its local minimum. The algorithm terminated after 2000 iterations after which the cost difference was less than 10^{-6} .

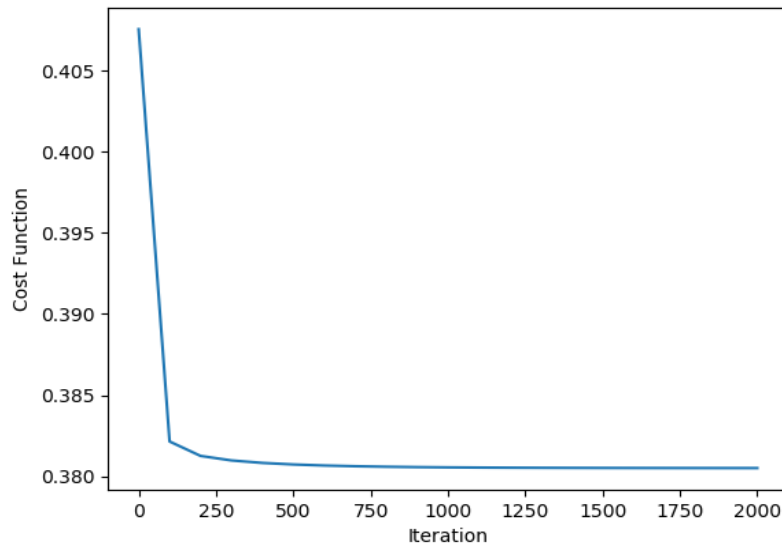


FIGURE 7.2: Cost Function

The output of the logistic regression model is the probability of 10 year risk of a cardiovascular heart disease (CHD). Different values of threshold were tried and then the threshold was set to 0.5 based on the accuracy of the model such that if $P(10 \text{ Year CHD}) < 0.5$ then the result is '0' else the result is '1'.

TABLE 7.1: Accuracy of the Logistic Regression Model

THRESHOLD	ACCURACY	SENSITIVITY	SPECIFICITY
0.1	53.5%	81.58%	48.41%
0.2	75.2%	48.25%	80.1%
0.3	82.61%	25.0%	93.07%
0.4	84.77%	10.53%	98.25%
0.5	84.97%	4.39%	99.6%

Therefore the logistic regression model was built using the Framingham heart study dataset [8] which produced the following results. The model has an

accuracy of approximately 85% with a sensitivity of 4.4% and specificity of 99.5%.

```

ACCURACY :
-----
True Negative: 1251
False Positive: 5
False Negative: 218
True Positive: 10
CORRECT PREDICTIONS : 84.97 %

```

FIGURE 7.3: Confusion Matrix

7.2 Paillier Encryption and Decryption

The data is encrypted using the Paillier cryptosystem. We have generated 16-bit public and private key pairs. The key pair is generated automatically when the client signs up into the cloud application.

Given a public key K_p , the message m can be encrypted as follows :

$$c = E(m, r) = (1 + mK_p)r^{K_p} \bmod K_p$$

where r is a random integer, $0 < r < K_p$

In order to incorporate the proxy re-encryption scheme into the Paillier cryptosystem, the generation of the random integer r must be controlled as explained in the previous chapter on proxy re-encryption. Therefore the encryption is slightly modified. The decryption however is implemented

unmodified. Given a public key K_p and private key K_s , the cipher text c can be decrypted as follows :

$$m = \frac{(c^{K_s} - 1)K_s^{-1} \bmod K_p^2}{K_p} \bmod K_p$$

```
C:\Users\USER\Documents\Final Year Project\Predictive Model>python Paillier.py
```

```
GENERATING KEYPAIRS
```

```
-----  
<PrivateKey: 23400 11506>    <PublicKey: 23707>
```

```
ENCRYPTING TWO NUMBERS
```

```
-----  
Plain Text X = 22  
Cipher Text of X, CX = 412659360
```

```
Plain Text Y = 6  
Cipher Text of Y, CY = 441565379
```

```
HOMOMORPHIC ADDITION
```

```
-----  
X + Y = Z = 28  
CX + CY = CZ = 382054797
```

```
Decrypting CZ :  
Plain text Z = 28
```

```
HOMOMORPHIC MULTIPLICATION
```

```
-----  
(Z + 2) * 3 = A = 90  
(CZ + 2) * 3 = CA = 152411547
```

```
Decrypting CA :  
Plain text A = 90
```

FIGURE 7.4: Paillier Cryptosystem Example

7.3 Prediction on Encrypted Data

The logistic regression model built in the previous section was used to analyze data in both plain text and encrypted form.

Example 1 (Positive outcome): For the following data - Gender = Female, Age = 60, BP Medication = No, Prevalent Stroke = No, Prevalent Hypertension = No, Diabetes = No, Total Cholestrol = 195 mg/dL, Systolic Blood Pressure = 106 mmHg, Diabolic Blood Pressure = 70 mmHg, BMI = 26.97, Glucose Level = 77 mg/dL, the model produce similar results for both plain and encrypted data.

```
C:\Users\USER\Documents\Final Year Project\Predictive Model>python PlainTextAnalysis.py
```

```
PLAIN TEXT ANALYSIS
```

```
-----
```

```
ORIGINAL DATA :
```

```
-----
```

```
[0, 60, 0, 0, 0, 0, 195, 106, 70, 26.97, 77]
```

```
-----
```

```
Probability : 0.7502651555149078
```

```
Prediction : 1
```

```
-----
```

FIGURE 7.5: Plain Text Analysis - Positive Outcome

```
C:\Users\USER\Documents\Final Year Project\Predictive Model>python CipherTextAnalysis.py
```

```
CIPHER TEXT ANALYSIS
```

```
-----
```

```
ORIGINAL DATA :
```

```
-----
```

```
[0, 60, 0, 0, 0, 0, 195, 106, 70, 26.97, 77]
```

```
GENERATING KEYPAIRS
```

```
-----
```

```
<PrivateKey Ks: 40228> <PublicKey: 40633>
```

```
ENCRYPTED DATA :
```

```
-----
```

```
[163692070, 863124906, 893142226, 461114079, 952917018, 814694116, 817645298, 1333289749, 1499516311, 1179831472, 918492326]
```

```
-----
```

```
Probability : 0.7500726888282483
```

```
Prediction : 1
```

```
-----
```

FIGURE 7.6: Cipher Text Analysis - Positive Outcome

Example 2 (Negative outcome): For the following data - Gender = Female, Age = 20, BP Medication = No, Prevalent Stroke = No, Prevalent Hypertension = No, Diabetes = No, Total Cholestrol = 313 mg/dL, Systolic Blood Pressure = 100 mmHg, Diabolic Blood Pressure = 71 mmHg, BMI = 21.68, Glucose Level = 78 mg/dL, the model produce similar results for both plain and encrypted data.

```
C:\Users\USER\Documents\Final Year Project\Predictive Model>python PlainTextAnalysis.py
```

```
PLAIN TEXT ANALYSIS
```

```
-----  
ORIGINAL DATA :
```

```
-----  
[0, 20, 0, 0, 0, 0, 313, 100, 71, 21.68, 78]
```

```
-----  
Probability : 0.48534346738428247
```

```
Prediction : 0  
-----
```

FIGURE 7.7: Plain Text Analysis - Negative Outcome

```
C:\Users\USER\Documents\Final Year Project\Predictive Model>python CipherTextAnalysis.py
```

```
CIPHER TEXT ANALYSIS
```

```
-----  
ORIGINAL DATA :
```

```
-----  
[0, 20, 0, 0, 0, 0, 313, 100, 71, 21.68, 78]
```

```
GENERATING KEYPAIRS
```

```
-----  
<PrivateKey Ks: 53788> <PublicKey: 54253>
```

```
ENCRYPTED DATA :
```

```
-----  
[2413190886, 2348502002, 917424061, 2688396685, 1077372023, 104609014, 1701178855, 1661569106, 1102447381, 1054059140, 1  
616482443]
```

```
-----  
Probability : 0.48557900075205973
```

```
Prediction : 0  
-----
```

FIGURE 7.8: Cipher Text Analysis - Negative Outcome

7.4 Proxy Re-encryption between Two Users

The proxy re-encryption scheme for the Paillier cryptosystem has been successfully implemented allowing users to share encrypted data based on a shared seed value.

The value of the seed that is shared between the two users must be much larger than the plain text that is to be shared. After generating the seed value, the encrypted plain text is subtracted from the encrypted seed by the owner of the data to generate the difference based on the formula described in the previous section.

The following figure contains the code snippet in Python used in proxy re-encryption for a Paillier Cryptosystem.

It is based on the equation

$$m_1 - m_2 = \frac{E(m_1, r)E(m_2, r)^{-1} - 1 \bmod K_p^2}{K_p} \bmod K_p$$

```
inverse_cipher_a = invmod(cipher_text_a, pub_a.n_sq)
step_1 = (encrypt_x0 * inverse_cipher_a) % pub_a.n_sq
step_2 = (step_1 - 1) % pub_a.n_sq
step_3 = (step_2 // pub_a.n) % pub_a.n
difference = step_3
```

FIGURE 7.9: Code Snippet for Computing the Difference between two Paillier Encrypted Numbers

In the above code snippet, the object *pub_a* denotes the public key of a user A. The object *cipher_text_a* denotes the cipher text that is to be passed through the proxy re-encryption scheme. The object *encrypt_x0* denotes the encrypted form of the secret seed that is established between user A and another user B. The function *invmod()* is used to compute the inverse of the cipher text modulo *pub_a*. The object *difference* corresponds to the result $m_1 - m_2$ in the above equation. The object *difference* is then normally encrypted using the public key of user B.

The code snippet is more conveniently demonstrated using the following example. It involves the passing of the plain text ‘61’ from a user A to user B by establishing a seed ‘116’ between them. The seed is generated such that it is much larger than the plaintext.

GENERATING KEYPAIRS

User A :

<PrivateKey Ks: 23392> <PublicKey Kp: 23701>

User B :

<PrivateKey Ks: 42840> <PublicKey Kp: 43259>

GENERATING THE PLAIN TEXT

Plain Text : 61

GENERATING THE RANDOM INTEGER r TO BE USED FOR ENCRYPTION

Random Integer r : 23399

ENCRYPTING MESSAGE USING PUBLIC KEY OF USER A AND r

Cipher Text : 488666565

GENERATING SEED X_0 BETWEEN USER A AND USER B

Seed X_0 : 116

ENCRYPTING SEED X_0 USING PUBLIC KEY OF USER A AND r

Encrypted Seed : 199182551

COMPUTING THE DIFFERENCE BETWEEN THE ENCRYPTED SEED X_0 AND CIPHER TEXT

The difference between plain text and X_0 : 55

ENCRYPTING THE DIFFERENCE USING PUBLIC KEY OF USER B

Cipher Text Difference : 305173928

DECRYPTING THE DIFFERENCE USING PRIVATE KEY OF USER B

Plain Text Difference : 55

SUBTRACTING THE SEED X_0 FROM THE DECRYPTED DIFFERENCE

Plain Text : 61

FIGURE 7.10: Proxy Re-encryption Example

7.5 Deploying the Application

The project involved the development of two applications.

1. A Client side application that performed the following :
 - Encryption of sensitive client data.
 - Decryption of the response from the server application.
2. A server side cloud application that performed the following :
 - Predictive analysis.
 - Proxy re-encryption.

Both the applications were developed using the Python Django framework. The server side cloud application was deployed on the Heroku cloud platform. The cloud application is accessible at the url '<https://cloud-fyp.herokuapp.com/>' while the client application is run on the local Django server. Data between the two applications is communicated in the JSON format which is achieved with help from the Django REST framework. The cloud application maintains an SQLite database that stores the authentication information, the authorization information for proxy re-encryption and the encrypted result of the analysis for later retrieval.

Figure 7.11 illustrates the flow of data between the two application. Figures 7.12 and 7.13 show the directory structure of the cloud service and the client application respectively. Figures 7.14 to 7.19 demonstrate a working example of the system.

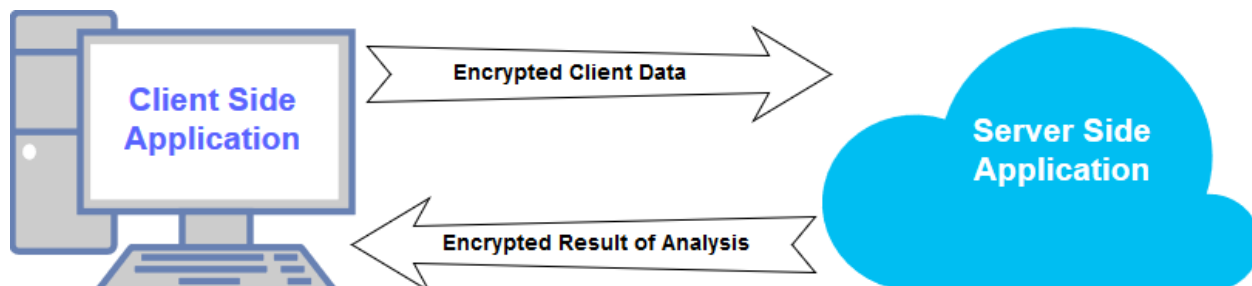


FIGURE 7.11: Flow of Data

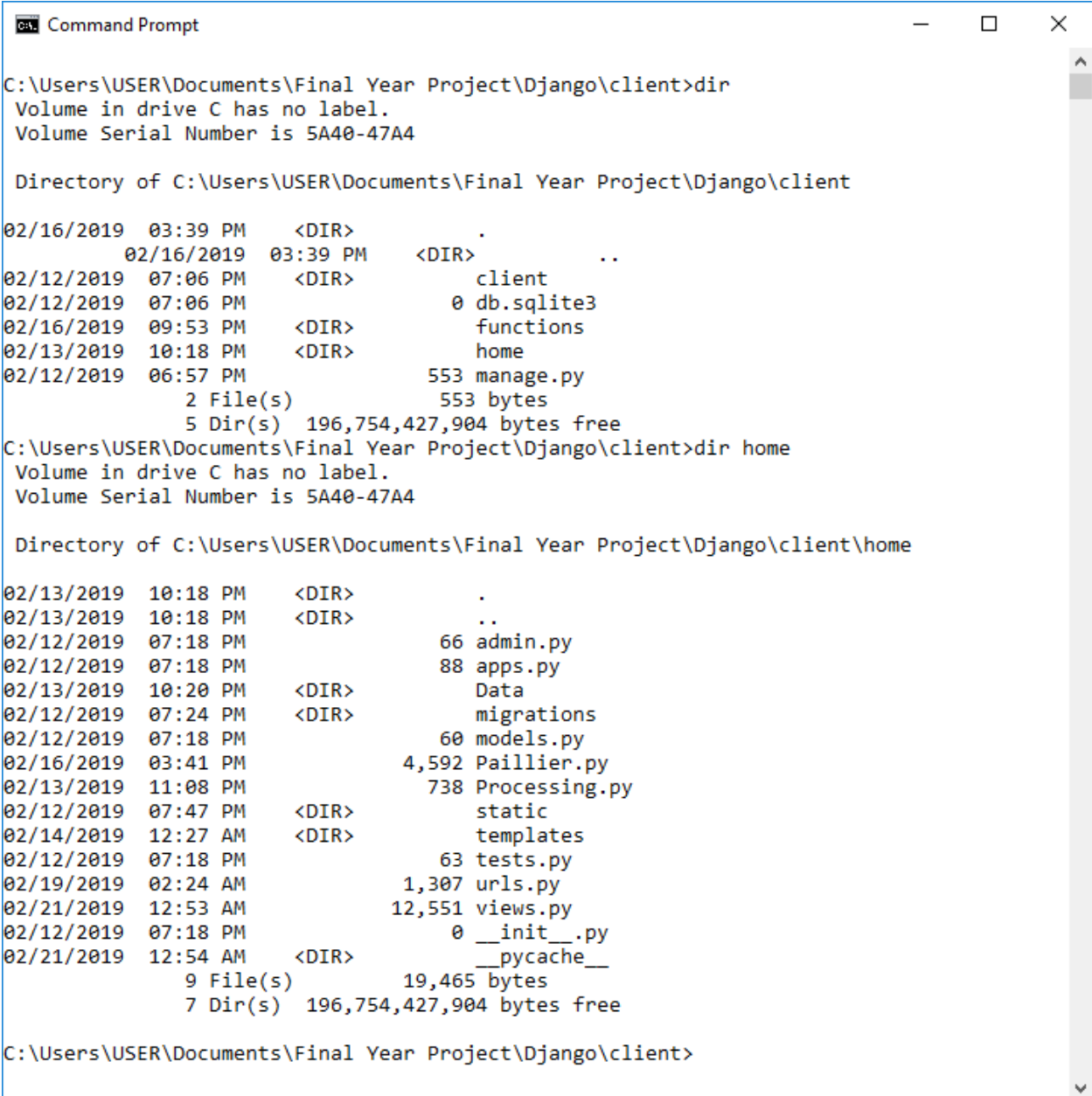
```

Command Prompt - heroku run bash

(Heroku) C:\Users\USER\Documents\Final Year Project\Heroku\cloud>heroku run bash
Running bash on cloud-fyp... up, run.1978 (Free)
~ $ ls -a
. analysis data_models .heroku manage.py Procfile requirements.txt signup
.. cloud db.sqlite3 login manage_result .profile.d runtime.txt staticfiles
~ $ ls analysis
admin.py apps.py __init__.py LogisticRegression migrations models.py __pycache__ tests.py
urls.py views.py
~ $ ls data_models
admin.py Data LogisticRegression.py models.py Processing.py RegressionModelling.py
tests.py views.py
apps.py __init__.py migrations Paillier.py __pycache__ serializers.py
urls.py
~ $ ls signup
admin.py apps.py __init__.py migrations models.py Paillier.py __pycache__ static tests.py
urls.py views.py
~ $ ls cloud
__init__.py __pycache__ settings.py urls.py wsgi.py
~ $ ls login
admin.py apps.py __init__.py migrations models.py __pycache__ tests.py urls.py views.py
~ $ ls manage_result
admin.py apps.py __init__.py migrations models.py __pycache__ tests.py urls.py views.py
~ $

```

FIGURE 7.12: Directory Structure of the Cloud Service on Heroku



```

C:\Users\USER\Documents\Final Year Project\Django\client>dir
Volume in drive C has no label.
Volume Serial Number is 5A40-47A4

Directory of C:\Users\USER\Documents\Final Year Project\Django\client

02/16/2019  03:39 PM    <DIR>          .
               02/16/2019  03:39 PM    <DIR>          ..
02/12/2019  07:06 PM    <DIR>          client
02/12/2019  07:06 PM                0 db.sqlite3
02/16/2019  09:53 PM    <DIR>          functions
02/13/2019  10:18 PM    <DIR>          home
02/12/2019  06:57 PM                553 manage.py
               2 File(s)                553 bytes
               5 Dir(s) 196,754,427,904 bytes free
C:\Users\USER\Documents\Final Year Project\Django\client>dir home
Volume in drive C has no label.
Volume Serial Number is 5A40-47A4

Directory of C:\Users\USER\Documents\Final Year Project\Django\client\home

02/13/2019  10:18 PM    <DIR>          .
02/13/2019  10:18 PM    <DIR>          ..
02/12/2019  07:18 PM                66 admin.py
02/12/2019  07:18 PM                88 apps.py
02/13/2019  10:20 PM    <DIR>          Data
02/12/2019  07:24 PM    <DIR>          migrations
02/12/2019  07:18 PM                60 models.py
02/16/2019  03:41 PM            4,592 Paillier.py
02/13/2019  11:08 PM                738 Processing.py
02/12/2019  07:47 PM    <DIR>          static
02/14/2019  12:27 AM    <DIR>          templates
02/12/2019  07:18 PM                63 tests.py
02/19/2019  02:24 AM            1,307 urls.py
02/21/2019  12:53 AM           12,551 views.py
02/12/2019  07:18 PM                0 __init__.py
02/21/2019  12:54 AM    <DIR>          __pycache__
               9 File(s)           19,465 bytes
               7 Dir(s) 196,754,427,904 bytes free

C:\Users\USER\Documents\Final Year Project\Django\client>

```

FIGURE 7.13: Directory Structure of the Client Application on Django

Predictive Analysis on Encrypted Data with Multi-User Access - Client

Please Log In

Log In

Sign Up

FIGURE 7.14: Home Page

Predictive Analysis on Encrypted Data with Multi-User Access - Client

YOUR ACCOUNT HAS BEEN CREATED!

Your Account Details

Username	<input type="text" value="User1"/>
Private Key	<input type="text" value="25456"/>
Private Key	<input type="text" value="25777"/>

Continue

FIGURE 7.15: Account Confirmation

Predictive Analysis on Encrypted Data with Multi-User Access - Client

USER: User1

Analysis

Your Result

Access Result

Authorize

De-authorize

Sign Out

Enter the following data :

Gender :	<input type="radio"/> Male <input checked="" type="radio"/> Female	Total Cholestrol (mg/dL) :	<input type="text" value="195"/>
Age :	<input type="text" value="60"/>	Systolic Blood Pressure (mmHg) :	<input type="text" value="106"/>
BP Medication :	<input type="radio"/> Yes <input checked="" type="radio"/> No	Diabolic Blood Pressure (mmHg) :	<input type="text" value="70"/>
History of Stroke :	<input type="radio"/> Yes <input checked="" type="radio"/> No	BMI (kg/m2) :	<input type="text" value="26.97"/>
Hypertension :	<input type="radio"/> Yes <input checked="" type="radio"/> No	Glucose (mg/dL) :	<input type="text" value="77"/>
Diabetes :	<input type="radio"/> Yes <input checked="" type="radio"/> No		

Encrypt Data

FIGURE 7.16: Client Data

Predictive Analysis on Encrypted Data with Multi-User Access - Client

USER: User1

Analysis

Your Result

Access Result

Authorize

De-authorize

Sign Out

Your Encrypted Data :

Gender :	<input type="text" value="385331719"/>	Total Cholestrol (mg/dL) :	<input type="text" value="97737353"/>
Age :	<input type="text" value="174163790"/>	Systolic Blood Pressure (mmHg) :	<input type="text" value="653239023"/>
BP Medication :	<input type="text" value="512061993"/>	Diabolic Blood Pressure (mmHg) :	<input type="text" value="523316518"/>
History of Stroke :	<input type="text" value="491428214"/>	BMI (kg/m2) :	<input type="text" value="22657972"/>
Hypertension :	<input type="text" value="433221235"/>	Glucose (mg/dL) :	<input type="text" value="541155225"/>
Diabetes :	<input type="text" value="404684291"/>		

Analysis

FIGURE 7.17: Encrypted Client Data

Predictive Analysis on Encrypted Data with Multi-User Access - Client

USER : User1

Analysis

Your Result

Access Result

Authorize

De-authorize

Sign Out

Your Encrypted Result

Cipher Result

212840586

Decrypt

FIGURE 7.18: Encrypted Result of Analysis

Predictive Analysis on Encrypted Data with Multi-User Access - Client

USER : User1

Analysis

Your Result

Access Result

Authorize

De-authorize

Sign Out

Your Analysis Result

Prediction Probability :

0.7500726888282483

Risk of 10 Year CHD :

Yes

Done

FIGURE 7.19: Decrypted Result of Analysis

CHAPTER 8

CONCLUSION AND FUTURE WORK

Homomorphic encryption allows us to encrypt data, perform calculations with it such that when the result is decrypted, the resulting plain text is equivalent to the result obtained by performing the same calculations on the original data instead of the encrypted data. The advantage of this feature is that unreliable parties can perform computation on sensitive data. This provides clients the ability to extract the computing power of entities that may or may not be reliable, such as the publicly available cloud computing service that has been implemented as a part of our proposed system. Our proposed system demonstrates the use of a homomorphic encryption scheme (Paillier cryptosystem) in the computation involved in the predictive analysis algorithm, logistic regression.

The predictive analysis service operates in the cloud. The client encrypts his/her private data and sends it to the cloud service. After completion of the analysis, the results are available in encrypted form for retrieval from the cloud. The owner of the data can simply retrieve the encrypted result and decrypt it. Other users who are authorized to access the results can retrieve the results with the proxy re-encryption mechanism.

Additionally, the cloud service provides authorized third parties the ability to access the encrypted results, which is otherwise only accessible only by the client (owner of the data). This has been made feasible with the use of a proxy re-encryption scheme. The cloud service acts as the proxy between the data owner and the third party. The technique used in this implementation does not suffer from any overhead, as it requires only one extra level of encryption.

Future work can involve extending the use of homomorphic encryption schemes to other analysis algorithms. A partial homomorphic encryption (PHE) scheme is homomorphic with respect to only one operation, that is, addition or multiplication. A somewhat homomorphic encryption (SHE) is more general than PHE as it supports both addition and multiplication. A fully homomorphic encryption scheme (FHE) allows any arbitrary operation. A PHE is more efficient than an FHE, but it is limited in its capability. Therefore FHE schemes can be explored to apply it suitably in different algorithms.

REFERENCES

1. Chen Guo, Xingbing Fu, Yaojun Mao, Guohua Wu, Fagen Li and Ting Wu (2018) ‘Multi-User Searchable Symmetric Encryption with Dynamic Updates for Cloud Computing’, MDPI, Vol. 9, Issue 10, 242.
2. Doron Shaked, Omer Barkol (2017) ‘Data Analytics on Encrypted Data Elements’, United States Patent Application Publication, Pub. No.: US 2017/0170960 A1.
3. Joppe W. Bos, Kristin Lauter, Michael Naehrig (2014) ‘Private predictive analysis on encrypted medical data’, Journal of Biomedical Informatics, Vol. 50, pp. 234–243.
4. Reda Bellafqira, Gouenou Coatrieux, Dalel Bouslimi, Gwenole Quellec and Michel Cozic (2017) ‘Sharing Data Homomorphically Encrypted with Different Encryption Keys’, Cornell University, arXiv:1706.01756v1.
5. Susan R. Hohenberger, Kevin Fu, Giuseppe Ateniese, Matthew Green (2008) ‘Unidirectional Proxy Re-encryption’, United States Patent Application Publication, Pub. No.: US 2008/0059787 A1.
6. Wenxiu Ding, Zheng Yan, Robert H. Deng (2017) ‘Encrypted data processing with Homomorphic Re-Encryption’, Information Sciences, Vol. 409-410, pp. 35-55.
7. Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, Lihua Wang, National Institute of Information and Communications Technology (NICT), Japan (2016) ‘Scalable and Secure Logistic Regression via Homomorphic Encryption’, CODASPY 2016, pp. 142-144.

8. <https://www.kaggle.com/amanajmera1/framingham-heart-study-dataset/version/1> 'Framingham Heart Study Dataset'.
9. https://en.wikipedia.org/wiki/Homomorphic_encryption 'Homomorphic Encryption'.
10. https://en.wikipedia.org/wiki/Logistic_regression 'Logistic Regression'.
11. https://en.wikipedia.org/wiki/Paillier_cryptosystem 'Paillier Cryptosystem'.
12. https://en.wikipedia.org/wiki/Proxy_re-encryption 'Proxy re-encryption'.