11-21-2019

Privacy-Preserving Statistical Analysis of Health Data Using Paillier Homomorphic Encryption and Permissioned Blockchain

Mahdi Ghadamyari
University of Windsor

Privacy-Preserving Statistical Analysis of Health Data Using Paillier Homomorphic Encryption and Permissioned Blockchain

By

Mahdi Ghadamyari

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2019

Privacy-Preserving Statistical Analysis of Health Data Using Paillier Homomorphic Encryption and Permissioned Blockchain

by

Mahdi Ghadamyari

APPROVED BY:

M. Hlynka
Department of Mathematics and Statistics

J. Lu
School of Computer Science

S. Samet, Advisor
School of Computer Science

November 14, 2019

DECLARATION OF ORIGINALITY

1. Co-Authorship

I hereby declare that this thesis incorporates material that is the result of research conducted under the supervision of Dr. Saeed Samet (Advisor). In all cases, the key ideas, primary contributions, experimental designs, data analysis and interpretation, were performed by the author, and the contribution of co-author was primarily through the proofreading of the published manuscripts.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

2. Previous Publication

This thesis includes two original papers that has been previously submitted in peer reviewed conferences, as follows:

Chapter	Full Citation	Publication Status
All Chapters	Mahdi Ghadamyari, Dr. Saeed Samet. "Privacy-Preserving Statistical Analysis of Health Data Using Paillier Homomorphic Encryption and Permissioned Blockchain". The First Workshop on Security and Privacy on Blockchain for Big Data Applications (SPB 2019) In conjunction with 2019 IEEE International Conference on Big Data, Dec 9-12, Los Angeles, CA	Accepted
1 and 2	Mahdi Ghadamyari, Dr. Saeed Samet. "DEHR: A Consortium Blockchain Prototype for Managing Electronic Health Records". 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)	Submitted

3. General

I certify that to the best of my knowledge my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Blockchain is a decentralized and peer-to-peer ledger technology that adds transparency, traceability, and immutability to data. It has shown great promise in mitigating the interoperability problem and privacy concerns in the de facto electronic health record management systems and has recently received increasing attention from the healthcare industry. Several blockchain-based and decentralized health data management mechanisms have been proposed to improve the quality of care delivery to patients. Apart from care delivery, health data has other important applications, such as education, regulation, research, public health improvement, and policy support. However, existing privacy acts prohibit health institutions and providers from sharing patients' data with third parties. Therefore, research institutions that conduct research on private health data need a secure system that provides accurate analysis results while preserving patient privacy and minimizing the risks of data breaches. In this thesis, We propose a novel privacy-preserving method for statistical analysis of health data. We leveraged the blockchain technology and Paillier encryption algorithm to increase the accuracy of data analysis while preserving the privacy of patients. Smart contracts were used to carry out mathematical operations on the encrypted records in a secure manner. We were able to successfully deploy the proposed scheme on Hyperledger Fabric, a permissioned and consortium blockchain platform. Compared to the previous works, the proposed model enjoys the benefits of a distributed blockchain-based environment, which include higher availability and enhanced data security. The experimental results show the feasibility of this method with a reasonable amount of time for regular queries.

DEDICATION

To My Beloved Parents

Mr. & Mrs. Hassan and Zahra Ghadamyari

And

To My Brothers

Ehsan, Mohsen and Mostafa

ACKNOWLEDGEMENTS

I would like to express my sincerest gratitude to my supervisor, Dr. Saeed Samet, for his immense supports throughout my graduate studies. Thanks to his plenty of guidance, encouragement, and advice. This thesis would not have been possible without his patience and valuable guidance.

I would like to thank my thesis committee members Dr. Lu and Dr. Hlynka, for taking the time to review my thesis and attending my thesis proposal and defense sessions. Thanks to their guidance and valuable suggestions for improving this thesis.

Lastly, I would like to thank my family and friends for all their supports and motivation over the years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Statistical analysis of health data is an important task in healthcare. However, existing healthcare systems are incompatible with this important need due to privacy restrictions. A recently emerged technology called Blockchain has shown a great promise mitigating this incompatibility. This thesis aims to improve existing statistical analysis protocols by leveraging the blockchain technology. We propose a novel method that enables researchers to conduct statistical analysis on health data in a privacy-preserving, secure and precise manner.

In this chapter, we review the terminology and main concepts we are using in this thesis. The concepts include Statistical Analysis, Permissioned Blockchain, and Paillier Cryptosystem.

1.1 Statistical Analysis

Statistical analysis is the science of collecting and interpreting numerical data for the purpose of identifying underlying patterns and making a more effective decision. It has many usages in healthcare [2]. Table 1.1.1 shows some example usages of statistical analysis in healthcare.

Entity	Usage
Researchers	Generating Reports, Research & Development
Government and Law makers	Regulation, Public Health Status Analysis
Physicians	Decision Supporting
Hospitals, Health Providers	Policy Making, Performance Improvement
Patients	Self health management

Table 1.1.1: Statistic Analysis Usages in Healthcare

1.2 Blockchain

Blockchain is the technology behind Bitcoin that enables this crypto-currency to validate transactions without the need for a trusted third-party.

1.2.1 Bitcoin

Bitcoin [39] is the first and largest decentralized digital currency and online payment system. It was introduced in 2009 by Satoshi Nakamoto and replaced central banks with computer nodes running worldwide to validate transactions. Bitcoin is based on proof, instead of trust, and operates without any trusted third-party in a fully distributed environment.

1.2.2 Blockchain Structure

Blockchain is a growing list of records (blocks) that are linked together using cryptography. Each block contains some data (such as financial transactions or medical records), a timestamp, hash of the previous block, and hash of its data. As an exception, the first block in a blockchain (Genesis Block) does not have a previous block hash.

Since blocks are linked using hash values, even a small change in the contents of one block changes the hash of that block and invalidates all the following blocks. Figure 1.2.3 illustrates the result of data manipulation in a blockchain data structure.

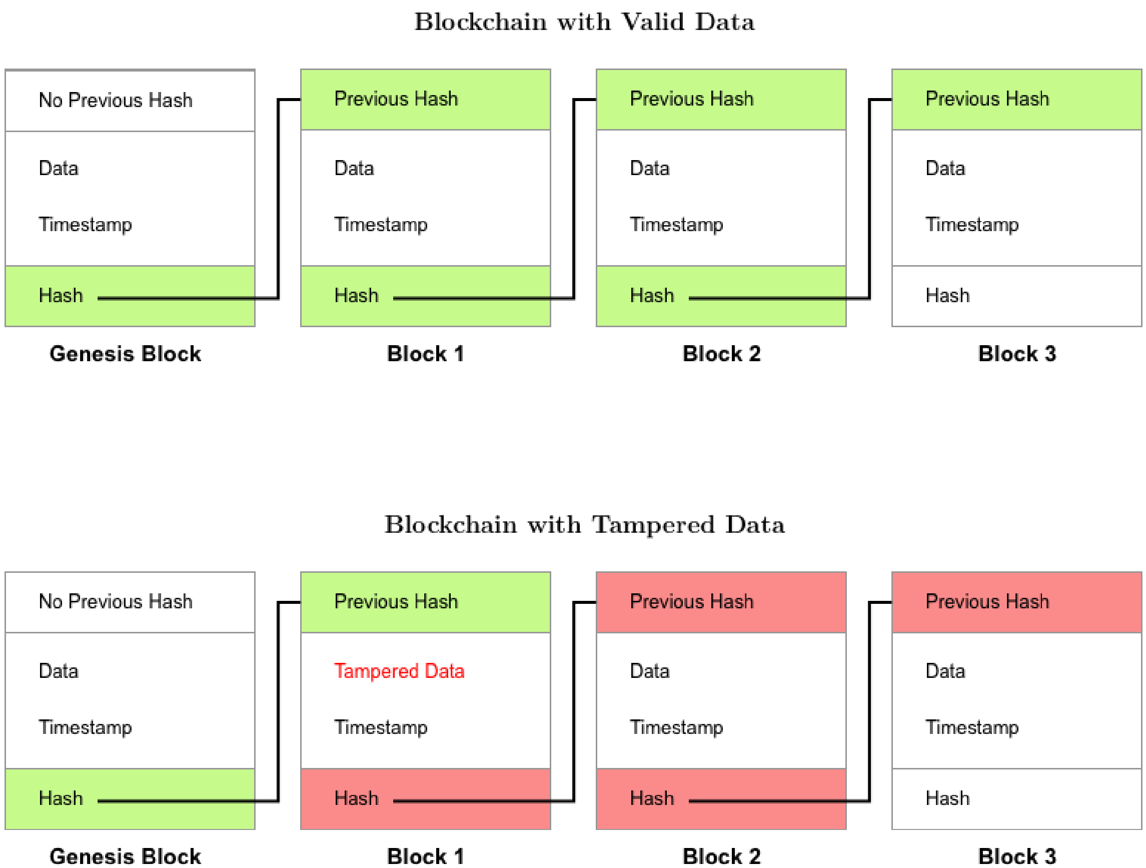

Fig. 1.2.1: Simple Blockchain Structures

The unique structure of blockchains provides some important features:

- **Distributed:** Every node in a blockchain network maintains a copy of the shared ledger. Therefore, there is not a single point of failure in the network, and the network becomes more stable.
- **Secure:** Every node validates data prior to adding it to the ledger, separately.
- **Immutable & Transparent:** The history of all data modifications will be permanently stored on the ledger, so records stored in a blockchain are transparent and traceable.

- **Programmable:** Computer programs (Smart Contracts) can be used to enforce terms of a contract in a blockchain network instead of trusted third-parties.

A blockchain is made of various technologies. Here we list the most critical technologies used in a blockchain network:

- **Node Connection:** Blockchain networks have a distributed architecture. In a blockchain network, nodes are connected using Peer to Peer communication protocols, e.g., BitTorrent [1].

- **Data Protection:** Hash algorithms like MD5 [43] are used to protect data against manipulation.

- **User Authentication and Transaction Validation:** Asymmetric cryptography and digital signature algorithms are utilized to authenticate users, and verify the integrity of transactions. Example algorithms are: RSA [45], and Elliptic Curve Cryptography [38].

- **Adding New Blocks:** Nodes in a blockchain network use consensus algorithms to reach a consensus on adding a new block to the blockchain. Example consensus algorithms used in blockchain networks are Proof of Work [56], Proof of Stake [55], Practical Byzantine Fault Tolerance [22].

1.2.3 Blockchain Types

Many derivations of blockchain technology have been introduced since its emergence in 2008. We can categorize them based on two factors:

- **Anonymity of Validators:** The nodes in a blockchain network that validate transactions are called validators. In a blockchain network, validators are either public or private. Blockchain networks with public validators allow public computers to join their network and validate transactions. On the other side, private or federated blockchain networks require computer nodes to obtain necessary certificates defined by the protocol before joining the network.

Bitcoin [39] and Ethereum [57] are two popular cryptocurrencies with a public blockchain structure and Hyperledger Fabric [15] is an example of a private blockchain.

- **Trust in Validators:** Trust in validators are either permissioned or permissionless. In a permissionless blockchain, the assumption is that everybody is potentially corrupt; therefore, the nodes use proof-based consensus algorithms instead of trust-based ones. However, proof-based consensus algorithms are time-consuming and consume a considerable amount of energy. On the other side, permissioned blockchains distribute the trust among a preselected set of participants to achieve a higher scalability rate. For example, Bitcoin is a permissionless blockchain, and Ethereum Casper is a permissioned blockchain.

Figure 1.2.2 illustrates various types of blockchains, examples of their consensus algorithms and their implementations categorized based on the two factors mentioned above.

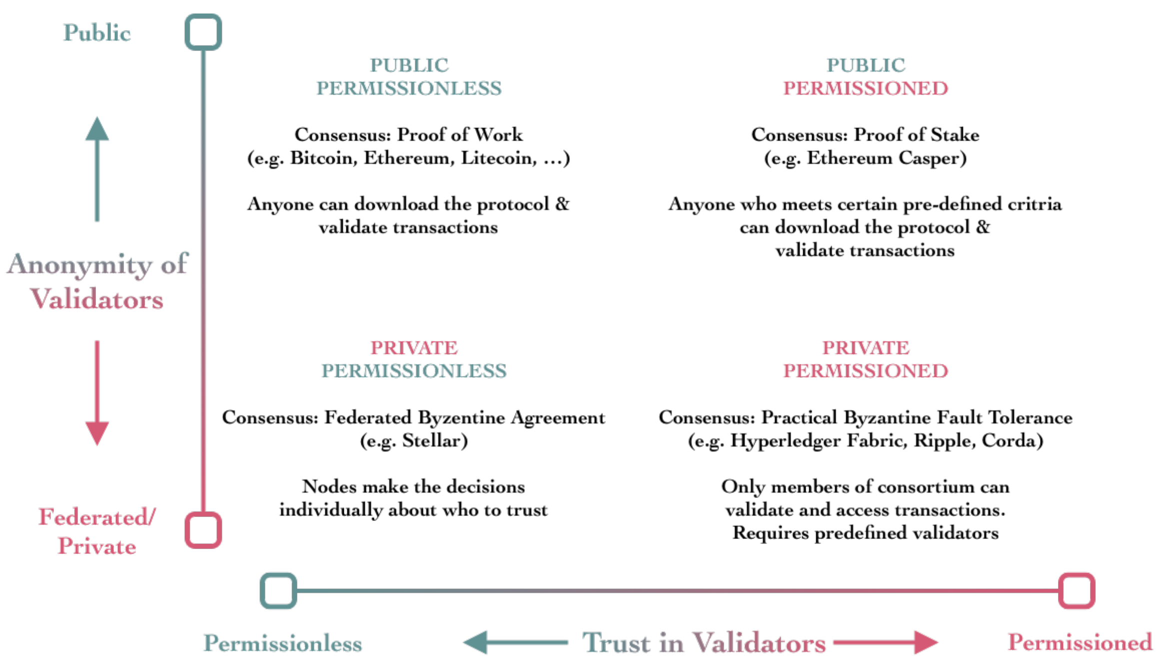

Fig. 1.2.2: Blockchain Types

A comparison of various blockchain types is provided in Table 1.2.1.

Category	Consensus	Users Anonymity	Ledger Immutability	Trust in Validators	Privacy	Scalability	Platforms
Permissionless and Public	PoW	High	High	None	Low	Low	Bitcoin
Permissioned and Public	PoS	High	Moderate	Low	Low	Moderate	Ethereum Casper, Ripple
Permissionless and Private	FBA	Moderate	Moderate	Moderate	High	Moderate	Holochain
Permissioned and Private	PBFT	Low	Low	High	High	High	Hyperledger

Table 1.2.1: Blockchain Types Comparison

1.2.4 Smart Contract

Smart Contracts are computerized versions of traditional contracts. They are a set of procedures defined by the blockchain network designer to process inputs and alter data stored on the distributed ledger once specific conditions defined by contract writers are met. With smart contracts, a computer program enforces the contract without the interference of any third-parties.
To create a smart contract, several parties agree on the terms of a contract. Then, they include the smart contract in a transaction and submit it to the blockchain network. Once the smart contract is added to the blockchain, the contract will be automatically triggered once specified terms in the contract are met.

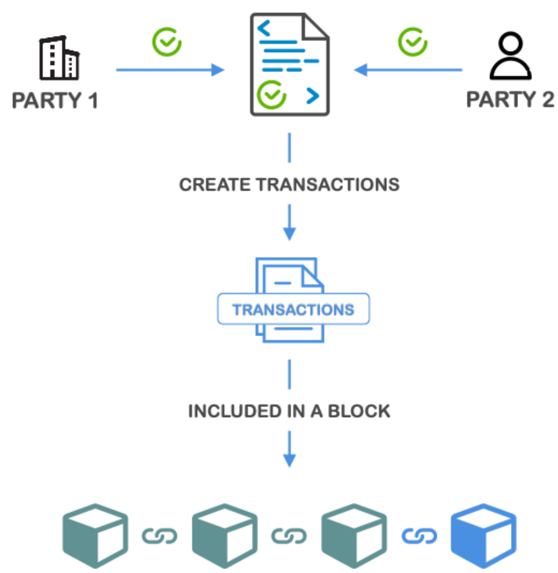

Fig. 1.2.3: A Smart Contract Submitted to A Blockchain Network

Some advantages of smart contracts are:

- Fraud reduction
- Arbitration and enforcement cost reduction
- Transaction cost reduction

1.2.5 Hyperledger

Hyperledger is an open-source collaborative effort from leaders in finance, banking, internet of things, supply chains, manufacturing, and technology for the goal of advancing cross-industry blockchain technologies and was first started in 2015. Linux Foundation hosts the project, but the project has received contributions from IBM, Intel, and SAP. Hyperledger incubates a range of business blockchain technologies, such as Hyperledger Fabric, Hyperledger Sawtooth, and Hyperledger Composer.

1.2.5.1 Hyperledger Fabric

Hyperledger Fabric is a modular and extensible open-source framework for deploying and operating private and permissioned blockchains. It provides confidentiality, flexibility, resiliency, and scalability and is one of the Hyperledger projects hosted by the Linux Foundation. Hyperledger Fabric supports smart contracts (also called chaincodes), configurable consensus, and membership services. It supports chaincodes written in Go and JavaScript language and has components for supporting other languages such as Java and Ethereum's Solidity language [5]. Due to its modular design, enterprise support, open-source environment, and support for several popular programming languages, it is potentially more flexible than its competitors like Corda and Quorum.

In Hyperledger Fabric, nodes can join multiple channels. Upon joining a channel, they receive an exact copy of the channel's ledger and continuously maintain the same view of the ledger. In this case, preserving the privacy participants in channels is a challenge. Particularly when the nodes are competitors or when they deal with sensitive data, e.g., health providers. We address this challenge using Paillier Cryptosystem, which will be described in the next section.

1.2.5.2 Hyperledger Composer

Hyperledger Composer is a set of tools for designing, deploying and testing business networks for Hyperledger Fabric. Hyperledger Composer is written in the JavaScript programming language and provides libraries for editors to have a convenient development environment. Hyperledger Composer abstracts away the complexity of creating business networks by offering a component-based solution in forms of a business network package (Figure 1.2.4).

A business network is consist of participants, assets, transactions, access control rules, events, and queries which are defined separately in several files:

- **Model File (.cto):** Modeling files written in Hyperledger Modeling Language [9] that contain definitions of all participants, assets, transactions, and events in the business network.

- **Transaction Logic (logic.js):** This file contains transaction processor functions that trigger whenever a transaction is called.

- **Access Control (permissions.acl):** A file that contains basic rules for controlling the access of users to the resources. Advanced access controls can be defined by adding a "permissionHelper.js" file to the package. With this additional file, advanced access to resources can be programmatically defined.

- **Query Definitions (queries.qry):** This file contains definitions of custom queries that are written in a SQL similar language (native query language). These queries can filter results returned using specified conditions and be triggered in transactions to perform operations such as updating or removing the results.

After a business network archive (.bna) file is created, the package can be deployed to Hyperledger Fabric instances or simulators to create a new or upgrade an existing business network.

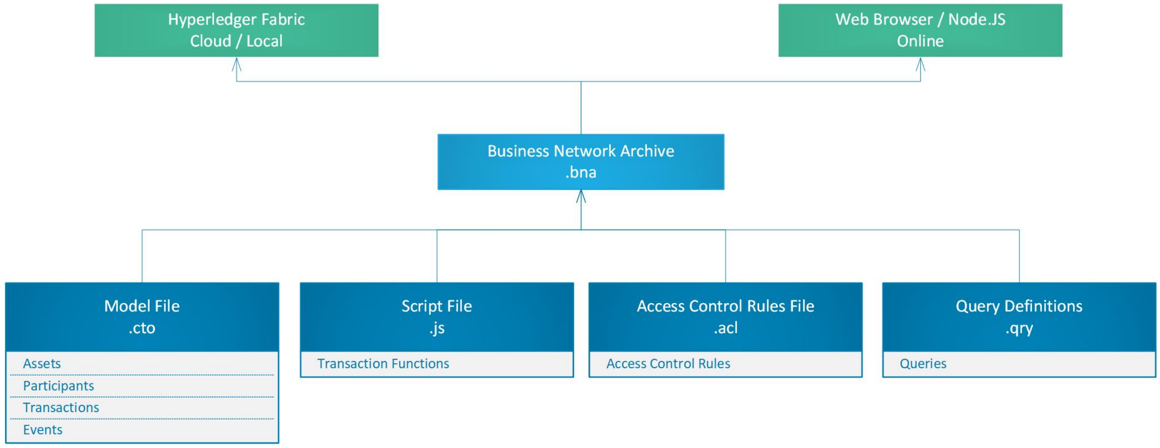

Fig. 1.2.4: Business Network Archive

1.3 Secure Multi-Party Computation (SMC)

Secure multi-party computation (SMC) refers to a process where multiple participants implement a joint computation without revealing information about the inputs and without the help of any trusted party.

1.4 Paillier Cryptosystem

Paillier cryptosystem [41] is an asymmetric homomorphic encryption algorithm introduced in 1999 by Pascal Paillier.
Asymmetric encryption algorithms have a pair of keys that contain a public key and a private key. Messages are encrypted using the public key and can only be decrypted using the private key (Figure 1.4.1).

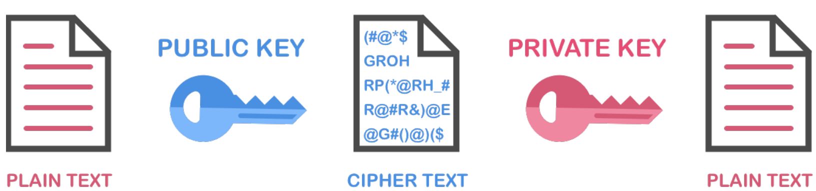

Fig. 1.4.1: Asymmetric encryption

Algorithm 1.4.1 shows steps for generating a Paillier cryptosystem.

Algorithm 1.4.1 Paillier Key Generation

Input: Prime numbers p, q, $s.t.$ $p \neq q$, and $gcd(pq, (p-1)(q-1)) = 1$
Output: Public key (n, g), and private key (λ, μ)
1: $n := pq$
2: $g :=$ A random number $s.t.$ $g \in \mathbb{Z}^*_{n^2}$
3: $\lambda := lcm(p-1, q-1)$
4: $\mu := (\frac{g^{\lambda-1} \mod n^2}{n})^{-1} \mod n$

After the generation of the keys, the encrypted form of plaintexts can be obtained using Algorithm 1.4.2.

Algorithm 1.4.2 Paillier Message Encryption

Input: Public key (n, g), Plaintext m where $0 \leq \; m \; < n$
Output: Ciphertext c
1: $r :=$ A random number $s.t.$ $0 < r < n$, and $r \in \mathbb{Z}^*_{n^2}$
2: $c := g^m \times r^n \mod n^2$

Finally, algorithm 1.4.3 decrypts an encrypted message.

Algorithm 1.4.3 Paillier Message Decryption

Input: Private key (λ, μ), Ciphertext c
Output: Plaintext m
1: $m := (\frac{c^{\lambda-1} \mod n^2}{n}) \times m \mod n$

Two notable features of Paillier cryptosystem are:

- **Probabilistic Encryption:** This feature refers to the generation of a different ciphertext for the same message every time the plaintext is encrypted.
- **Homomorphic Encryption:** Homomorphic encryption is a form of encryption that allows performing mathematical operations on plaintexts in their encrypted form. For example, in the Paillier cryptosystem, the addition of plaintexts can be achieved using their ciphertexts.

Using the homomorphic property of Paillier, the production of two ciphertexts decrypts to the summation of their corresponding plaintexts. Algorithm 1.4.4 shows the steps for calculating the homomorphic addition of two plaintexts.

Algorithm 1.4.4 Paillier Homomorphic Addition

Input: Public key (n, g), Encryption function E, Ciphertexts c_1, c_2 of plaintexts m_1, m_2, respectively.

Output: Encrypted summation of plaintexts: $m_1 + m_2$

1: $E(m_1 + m_2 \mod n) := c_1 \times c_2 \mod n^2$

CHAPTER 2

Related Works

2.1 Secure Multi-Party Computation

Secure multi-party computation (SMC) has been an active research area for several decades. Andrew Yao introduced this concept to the scientific community in 1982 in a problem that is known today as Yao's Millionaires' problem [58]. The problem discusses two millionaires who wish to find out who is richer without revealing their actual wealth. The solution involves the utilization of one-way functions in interactive communications between the parties. Later, a more generalized solution was introduced in another work by Yao in 1986 [59]. The work discussed the generation of a random integer $N = p.q$ such that its secret (p, q) is hidden from both parties individually but is recoverable jointly whenever needed. Yao also introduced workarounds for secure computations between two parties. Goldreich, Micali, and Wigderson followed Yao's works and introduced two, secure, multi-party computation methods in 1987 [29, 28]. Sheikh et al. [49] classified solutions for SMC problems into three categories: Randomization, Anonymization, and Cryptographic.

2.1.1 Randomization Methods

Randomization is another method for performing secure multi-party computations over private data. In this method, parties add random noises [54] or swap values [32, 42] in their original datasets and form distorted datasets in order to protect their private values. Randomization protocols deal with the trade-off between the precision of computations and the security of private values in a database. These protocols try to maximize the accuracy of computations over distorted datasets while preserving the confidentiality of private data. Cliftonet et al. [24] proposed a secure sum protocol that allows multiple parties to compute the sum of their private data while keeping the confidentiality of their private data. However, randomization protocols usually increase the size of datasets and decrease the precision of computation results.

2.1.2 Anonymization Methods

Data de-identification is another method commonly used for performing secure computations over private data. K-anonymity [52] is an example of this method, introduced by Sweeney in 2002. This method derives the data set D' from the original data set D in a way that, for any attributes a in D there are at least k instances in D'. An example of k-anonymity is shown in Table 2.1.1. However, anonymized or pseudo-anonymized databases are prone to social engineering attacks. Ashwin et al. [33] have identified two re-identification attacks against this method.

Table 2.1.1: K-anonymity Example

ID	Zip Code	Age	Nationality	Disease
1	E9A 0H7	19	Iranian	Heart Disease
2	E9A 0H1	16	Romanian	Diabetes
3	E9A 0D4	17	Chinese	Heart Disease
4	E9A 0H2	13	Japanese	Cancer
5	B3T 0H2	35	Brazilian	HIV
6	B3T 0T2	33	Romanian	HIV
7	B3T 0H1	31	Brazilian	HIV
8	B3T 0D8	37	Chinese	HIV
9	T1R 0H2	33	Iranian	Diabetes
10	T1R 0B5	43	Iranian	Heart Disease
11	T1R 0V2	53	Chinese	Cancer
12	T1R 0E8	44	Brazilian	Diabetes

(a) Private Dataset

ID	Zip Code	Age	Nationality	Disease
1	E9A ***	< 20	*	Heart Disease
2	E9A ***	< 20	*	Diabetes
3	E9A ***	< 20	*	Heart Disease
4	E9A ***	< 20	*	Cancer
5	B3T ***	3*	*	HIV
6	B3T ***	3*	*	HIV
7	B3T ***	3*	*	HIV
8	B3T ***	3*	*	HIV
9	T1R ***	≥ 30	*	Diabetes
10	T1R ***	≥ 30	*	Heart Disease
11	T1R ***	≥ 30	*	Cancer
12	T1R ***	≥ 30	*	Diabetes

(b) 4-Anonymity Dataset

2.1.3 Cryptographic Methods

In [20], the authors proposed a method for supporting private data in a Hyperledger Fabric channel. The proposed method requires modification of the underlying structure of Fabric's network for adding two new components. As a showcase, the authors implemented an auction application and stored encrypted reservations and bidding values privately on the ledger. Their results showed a 0.3 s transaction execution time. However, their method requires some clients to have access to the same private keys that peers use for data encryption, which may raise some security concerns and may not be suitable for the statistical analysis of health records. Compared to our work, we do not require any modification in the underlying Hyperledger Fabric structure and do not distribute the private key between the peers. Our method can be plugged into existing blockchain applications and used instantly. The authors in [48, 21, 46], proposed privacy-preserving techniques and protocols for securely computing statistical analysis methods. However, their proposed protocols are highly interactive and require many data exchanges between the participating parties. Our work is an attempt to reduce this complexity by using the blockchain technology.

2.2 Blockchain Adoption in Healthcare

The efficiency of health data management systems has a significant impact on patient care. However, existing health management systems suffer from lack of interoperability, expensive implementation, maintenance, and security vulnerabilities [50, 11]. Studies have shown that blockchain technology has the potential to mitigate many of these problems [31, 36]. A successful example is Estonia's healthcare system that uses blockchain to verify the integrity of medical records and access logs [4]. Following, we briefly review some of the research works related to blockchain adoption in health record management systems.

The authors in [17] proposed a blockchain-based and decentralized health records management system called MedRec. They used a public blockchain that incentives researchers to mine new blocks in exchange for getting access to anonymized medical data. The authors claimed that their proposed system increases transparency of medical records, stability of the network, and confidentiality of data. This work was later continued by the authors in [40]. The authors replaced miners with a network of trusted providers that participate in a proof of authority consensus mechanism. They used blockchain to store permission contracts. In their work, providers can join the network and grant patients, and other entities access to their databases using their credentials.

The authors in [37] used a federated and private blockchain to explore an auditable identity and access management framework for EHR systems. Evaluation of their system showed a size of 3.8 MB for initialization of the blockchain with 2-3 seconds mining time for new transactions.

The authors in [23] presented an integration of a cloud and blockchain storage scheme to manage PHR data. They used off-chain cloud storage for storing large amounts of medical data and a blockchain for indexing and securing them. In their work, patients are in control of their data. However, the interoperability of their system is not examined.

The authors in [34] propose a framework for secure multiparty computation that uses cloud computing and Paillier homomorphic encryption to protect the privacy of patients.

In [12], the authors proposed an interactive model for a blockchain-based PHR system. In the proposed system, smart contracts are utilized to collect patients' health records, and blockchain technology is used to make transactions immutable and traceable. The authors claimed that their approach encourages physicians to have more engagement with their patients outside clinics resulting in better care delivery.

CHAPTER 3

Methodology

In this chapter, the motivation and methodology will be discussed.

3.1 Motivation

Health data are sensitive information and have strict privacy rules. Privacy acts such as HIPPA [13] restrict access to patients' data sets and encourage healthcare providers to maintain isolated databases. As these databases grow, they become less efficient and secure, which makes health care services more expensive and less accessible to the public.

A survey [51] conducted by Deloitte in 2018 from 624 physicians shows that interoperability is the top demand from physicians as 62 percent said that interoperability in the current systems needs more improvements (Figure 3.1.1).

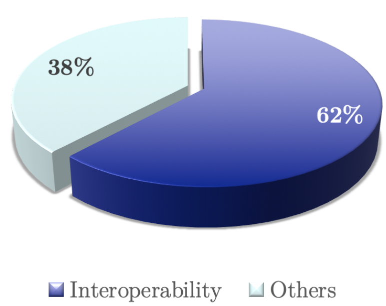

Fig. 3.1.1: Interoperability Is the Top Demand from Physicians

However, a recently emerged technology called Blockchain has shown great promise to mitigate this problem. Blockchain adoption in healthcare has received growing attention from industry as well as academia. In industry, a survey [53] conducted by

IBM from 200 executives shows 56% of participants expect to have a commercial blockchain solution at scale by 2020. 3.1.3

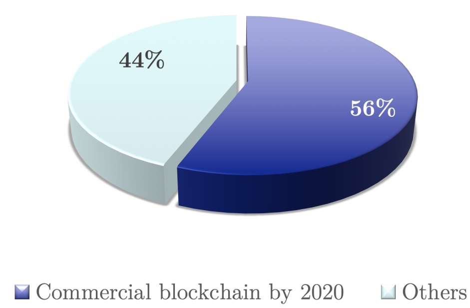

Fig. 3.1.2: Commercial Adoption of Blockchain in Healthcare Survey

In academia, publications related to blockchain adoption in healthcare increased from 5.56% in 2016 to 72.22% in 2018 [14].

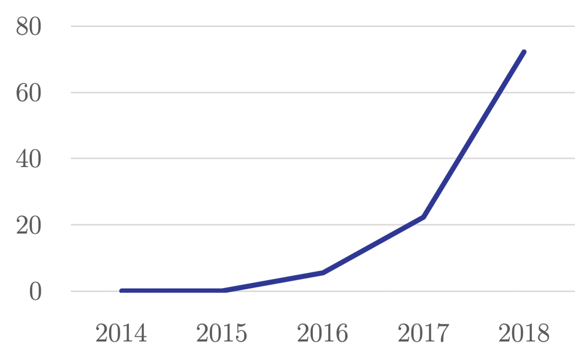

Fig. 3.1.3: Publications Related to Blockchain Adoption in Healthcare

Many works have been proposed for privacy-preserving statistical analysis on health data. However, they usually use a centralized solution to protect users' identities. The models have a low service quality due to having a single point of failure. On the other side, we saw that blockchain technology is expected to transform healthcare management systems in the near future. In addition, healthcare data custodians are often unable to provide researchers direct access to their data due to privacy concerns. So, there is a need for a secure statistical analysis protocol that is compatible with the characteristics of these new systems and provides researchers access to their desired data in blockchain networks.

In this work, we propose a privacy-preserving method to perform statistical analysis on health data in a distributed blockchain network.

3.2 Methodology

In our proposed framework, we use the blockchain technology to increase the transparency, accessibility, and integrity of the data and the Paillier cryptosystem to preserve the confidentiality of private data.
Our scheme is designed for a network with several data custodians and researchers. We assume data custodians are joined in a blockchain network channel and maintain a shared and distributed ledger. Additionally, researchers use APIs provided by the network for data communications (Figure 3.2.1).

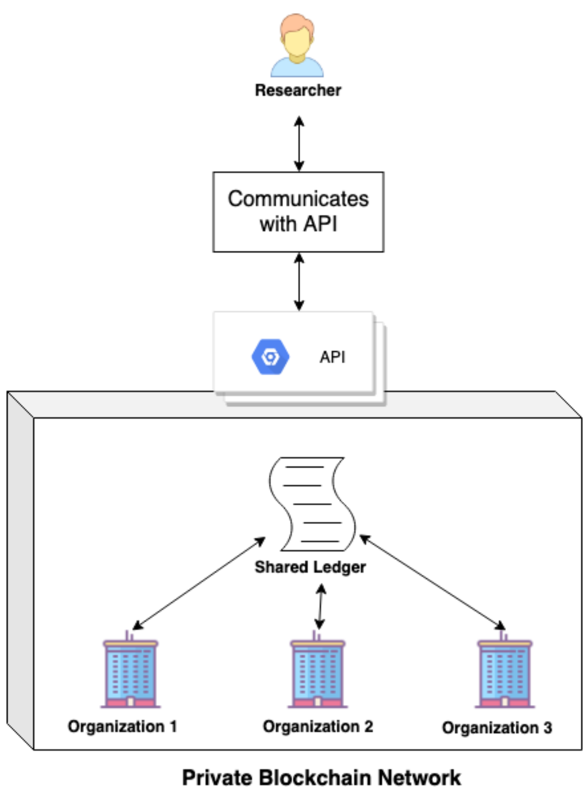

Fig. 3.2.1: A Sample Blockchain Network with 3 Organizations and 1 Researcher

The researchers send requests to the data custodians and ask them for the results of a specific query. The partial results of the query will be computed by each data custodian and encrypted using the Paillier cryptosystem. Smart contracts are utilized to compute the final result and preserve the privacy of the data. Data custodians can be any organizations, like health providers, insurance companies and etc. In a blockchain network, data custodians joined in the same channel are also maintainers of the ledger. Thus, they can read data stored on the ledger. On the other side, researchers are the end-users of the network, and their access to the ledger can be controlled using Access Control Lists (ACLs). In our scheme, the researcher is expected to not reveal any data to other parties.

3.2.1 Proposed Scheme

The method is consist of 6 steps (Figure 3.2.2) and proceeds as follows:

Step 1: The first step contains tasks that should be carried out by the researcher:

- Step 1.a: The researcher sets up a Paillier Cryptographic system with a private key SK_r and public key PK_r
- Step 1.b: The researcher stores the private key SK_r in a secure database
- Step 1.c: The researcher submits a proposal to the blockchain network for a new query. The proposal contains the description of the query, public key PK_r, and operation to be executed on the query results of data custodians to achieve the final result.

Step 2: A smart contract will create a new asset for the requested method.

Step 3: All data custodians will calculate the variables for the requested query, encrypt the values with the public key PK_r, and submit the encrypted values to the blockchain network.

Step 4: A smart contract will be executed to aggregate the encrypted variables and store the final result. This smart contract uses the homomorphic properties

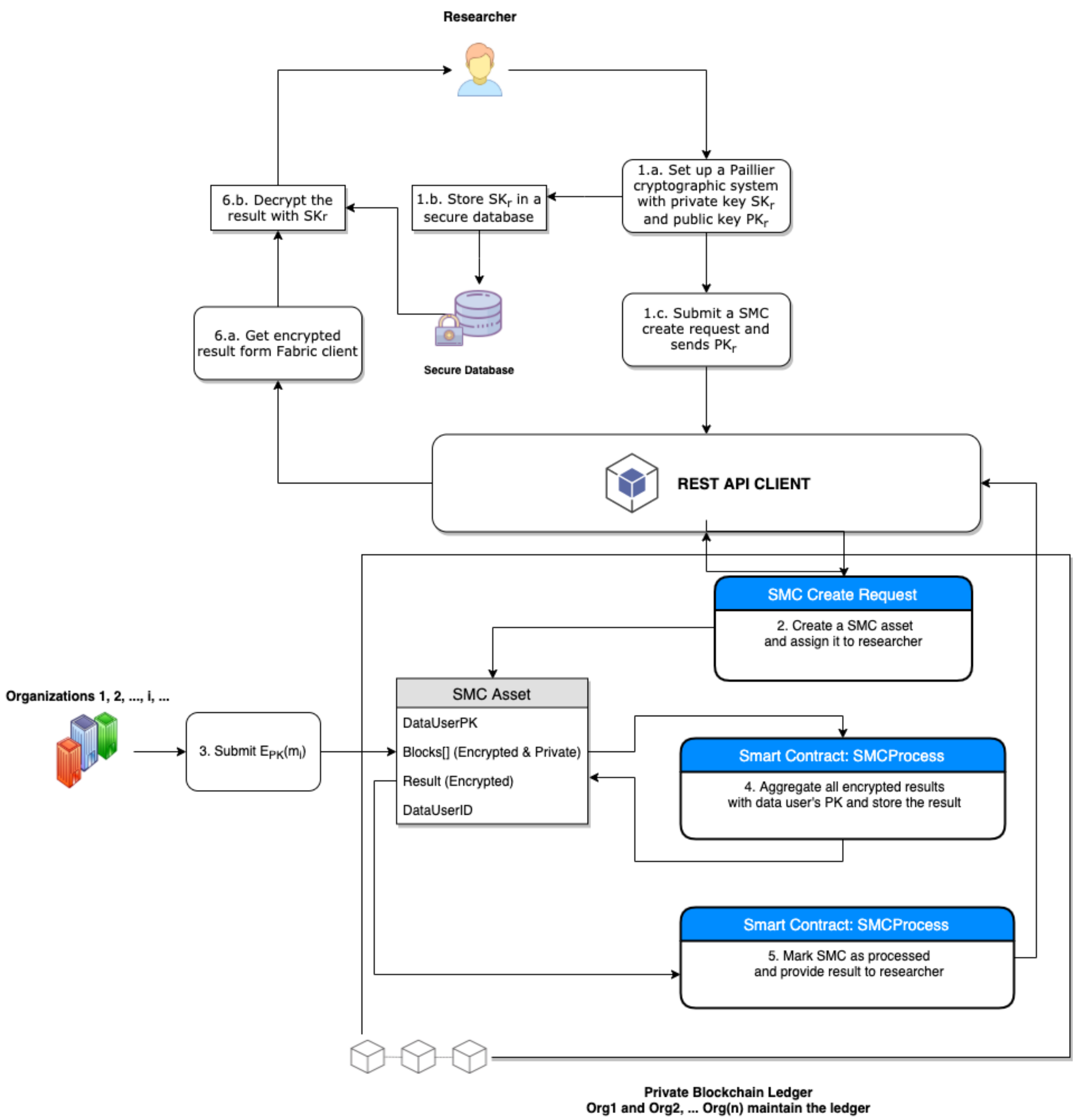

Fig. 3.2.2: Architecture of the Proposed Scheme

of the Paillier cryptosystem to calculate the final result. It will store the encrypted result on the blockchain that can be only decrypted using the private key of the researcher (SK_r).

Step 5: The final variables are ready and will be provided to the researcher.

Step 6: The researcher gets the values, decrypts them and uses them to calculate the associated statistical method function.

We demonstrated how the proposed scheme could be used to jointly calculate a statistical method and transfer its value to the researcher securely. All query results will be securely encrypted using the researcher's public key and then stored on the ledger. Therefore, encrypted data on the ledger is only decryptable by the researcher. Also, the researcher's access privileges can be restricted to specific values using ACLs, so the privacy of data custodians will be preserved.

Next, we show how some statistical functions can be securely calculated using the proposed method.

3.2.2 Secure Count

Count is a simple statistical function that represents the number of instances in a dataset.

Calculating count using our proposed method consists of two steps:

1. Each data owner calculates and encrypts the number of instances in their dataset and submits the result to the blockchain.

2. A smart contract will aggregate the encrypted values from each data owner and store the result on the blockchain.

3. The researchers receive the final value by decrypting the value obtained from the previous step.

Using these three simple steps, the data user receives the total number of instances without getting any knowledge about the actual number of instances within each organization. The count function has only one variable, which is the total number of instances. However, some statistical functions have more than one variable. A similar approach is used to calculate more functions with more calculations.

Table 3.2.1 demonstrates shares of data by each organization.

Data Owners	**Share**
D_1	$E(n_1)$
D_2	$E(n_2)$
$\dots$	$\dots$
D_i	$E(n_i)$
Homomorphic Addition:	**N**

Table 3.2.1: Shares of Data for Secure Count

3.2.3 Secure Mean

In a centralized dataset (Figure 3.2.2a), the mean value can be calculated using equation 1.

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{1}$$

In a distributed dataset, the equation 2 can be used for calculation of mean.

$$\overline{x} = \frac{\sum_{j=1}^{n_1} x_{1,j} + \sum_{j=1}^{n_2} x_{2,j} + \cdots + \sum_{j=1}^{n_i} x_{i,j}}{n_1 + n_2 + \cdots + n_i} \tag{2}$$

In our scheme, each party calculates, encrypts, and submits their shares to the blockchain network.

Table 3.2.2: Dataset Distribution Types

(a) Centralized Dataset

D_1
x_1
x_2
x_3
$\ldots$
x_n

(b) Distributed Dataset

D_1	D_2	$\ldots$	D_i
$x_{1,1}$	$x_{2,1}$	$\ldots$	$x_{i,1}$
$x_{1,2}$	$x_{2,2}$	$\ldots$	$x_{i,2}$
$\ldots$	$\ldots$	$\ldots$	$\ldots$
x_{1,n_1}	x_{2,n_2}	$\ldots$	x_{i,n_i}

Data Owners	**Shares**	
D_1	$E(\sum_{j=1}^{n_1} x_{1,j})$	$E(n_1)$
D_2	$E(\sum_{j=1}^{n_2} x_{2,j})$	$E(n_2)$
$\ldots$	$\ldots$	$\ldots$
D_i	$E(\sum_{j=1}^{n_i} x_{i,j})$	$E(n_i)$
Homomorphic Addition:	**A**	**N**

Shares of Data for Secure Mean

Then, smart contracts aggregate the shares, denoted with A (Equation 3), and N (Equation 4), using Paillier homomorphic properties and provide the aggregated results to the data user.

$$A = E(\sum_{j=1}^{n_1} x_{1,j}) \times E(\sum_{j=1}^{n_2} x_{2,j}) \times \cdots \times E(\sum_{j=1}^{n_i} x_{i,j}) \tag{3}$$

$$N = E(n_1) \times E(n_2) \times \cdots \times E(n_i) \tag{4}$$

Next, the data user decrypts the aggregated results and receives the sum of x and n

values. (Expression 5, and 6)

$$D(A) = \sum_{j=1}^{n_1} x_{1,j} + \sum_{j=1}^{n_2} x_{2,j} + \cdots + \sum_{j=1}^{n_i} x_{i,j} \tag{5}$$

$$D(N) = n_1 + n_2 + \cdots + n_i \tag{6}$$

Finally, the data user calculates the final mean value using equation 7.

$$\overline{x} = \frac{D(A)}{D(N)} \tag{7}$$

3.2.4 Secure Variance

In this section, we use a similar approach to the previous section to securely compute variance of a distributed dataset.

Equation 8 can be used to calculate variance in a centralized dataset.

$$v = \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n} \tag{8}$$

We expand this equation and change its form to derive equation 9:

$$v = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}(x_i^2 - 2x_i\overline{x} + \overline{x}^2)$$

$$= \frac{1}{n}(\sum_{i=1}^{n} x_i^2 - 2\overline{x}\sum_{i=1}^{n} x_i + n\overline{x}^2)$$

$$= \frac{\sum_{i=1}^{n} x_i^2}{n} - \frac{2}{n}\overline{x}\sum_{i=1}^{n} x_i + \overline{x}^2$$

$$= \frac{\sum_{i=1}^{n} x_i^2}{n} - \frac{2}{n}(\frac{1}{n}\sum_{i=1}^{n} x_i)\sum_{i=1}^{n} x_i + (\frac{1}{n}\sum_{i=1}^{n} x_i)^2$$

$$= \frac{\sum_{i=1}^{n} x_i^2}{n} - \frac{2}{n^2}(\sum_{i=1}^{n} x_i)^2 + \frac{1}{n^2}(\sum_{i=1}^{n} x_i)^2$$

$$= \frac{\sum_{i=1}^{n} x_i^2}{n} - \frac{(\sum_{i=1}^{n} x_i)^2}{n^2} \tag{9}$$

$$v = \frac{\sum_{j=1}^{n_1}(x_{1,j} - \overline{x_1})^2 + \sum_{j=1}^{n_2}(x_{2,j} - \overline{x_2})^2 + \cdots + \sum_{j=1}^{n_i}(x_{i,j} - \overline{x_i})^2}{n_1 + n_2 + \cdots + n_i} \tag{10}$$

Next, we define three variables, A, B, and N as shown in equations 11, 12, 13, respectively. Each organization calculates, encrypts and submits these variables to the blockchain, separately.

$$A = E(\sum_{i=1}^{n} x_i) \tag{11}$$

$$B = E(\sum_{i=1}^{n} x_i^2) \tag{12}$$

$$N = E(n) \tag{13}$$

Table 1 summarizes variables that i data owners (denoted with D) need to calculate and submit to blockchain (E denotes the encryption function of Paillier cryptosystem).

Data Owners	**Shares**		
D_1	$E(\sum_{j=1}^{n_1} x_{1,j})$	$E(\sum_{j=1}^{n_1} x_{1,j}^2)$	$E(n_1)$
D_2	$E(\sum_{j=1}^{n_2} x_{2,j})$	$E(\sum_{j=1}^{n_2} x_{2,j}^2)$	$E(n_2)$
...	...	...	...
D_i	$E(\sum_{j=1}^{n_i} x_{i,j})$	$E(\sum_{j=1}^{n_i} x_{i,j}^2)$	$E(n_i)$
Homomorphic Addition:	**A**	**B**	**N**

Shares of Data for Secure Variance

After all organizations submitted their values to blockchain, a smart contract will aggregate similar variables using Paillier homomorphic addition property and put the results on blockchain. Next, the data user reads and decrypts the value of variables from blockchain. Then, the data user calculates final value of variance using equation 14.

$$v = \frac{D(B)}{D(N)} - (\frac{D(A)}{D(N)})^2 \tag{14}$$

3.2.5 Secure Skewness

Skewness is a statistical method that shows the degree of assymetry of a distribution. Skewness can be calculated using expression 15 where σ is the standard deviation and μ is the mean value.

$$\gamma = \frac{\sum_{i=1}^{n}(x_i - \mu)^3}{n\sigma^3} \tag{15}$$

We expand this expression to change its form and make it suitable for a distributed environment as shown in expression 16.

$$\begin{aligned} \gamma &= \frac{1}{n\sigma^3}(\sum_{i=1}^{n}(x_i - \mu)^3) \\ &= \frac{1}{n\sigma^3}(\sum_{i=1}^{n}(x_i^3 - 3\mu x_i^2 + 3\mu^2 x_i - \mu^3)) \\ &= \frac{1}{n\sigma^3}(\sum_{i=1}^{n} x_i^3 - 3\mu \sum_{i=1}^{n} x_i^2 + 3\mu^2 \sum_{i=1}^{n} x_i - n\mu^3) \\ &= (\frac{1}{\sigma^3})(\frac{\sum_{i=1}^{n} x_i^3 - 3\mu \sum_{i=1}^{n} x_i^2}{n} + 2\mu^3) \end{aligned} \tag{16}$$

Based on expression 15, data owners need to calculate and securely share three values. Variables that data owners need to calculate are shown in table 3.2.3.

Data Owners	**Shares**		
D_1	$E(\sum_{j=1}^{n_1} x_{1,j}^2)$	$E(\sum_{j=1}^{n_1} x_{1,j}^3)$	$E(n_1)$
D_2	$E(\sum_{j=1}^{n_2} x_{2,j}^2)$	$E(\sum_{j=1}^{n_2} x_{2,j}^3)$	$E(n_2)$
...	...	...	...
D_i	$E(\sum_{j=1}^{n_i} x_{i,j}^2)$	$E(\sum_{j=1}^{n_i} x_{i,j}^3)$	$E(n_i)$
Homomorphic Addition:	**A**	**B**	**N**

Table 3.2.3: Shares of Data for Secure Skewness

Steps to securely calculate skewness are:

1. The data user securely calculates the mean (μ) and variance (σ^2) values based on the previous sections.
2. Data owners calculate the values of $\sum_{i=1}^{n} x_i^2$, $\sum_{i=1}^{n} x_i^3$, and n in their private datasets, encrypt the values and submit them to the blockchain network.
3. A smart contract securely aggregates the submitted values using Paillier Homomorphic addition property, and stores the final values (A, B and N) on the blockchain.
4. The data user recieves the final values, decrypts them and calculates the final value using expression 17.

$$\gamma = (\frac{1}{\sigma^3})(\frac{D(B) - 3D(A)}{D(N)} + 2\mu^3) \tag{17}$$

3.2.6 Other Methods

A similar approach as the previous schemes can be used to calculate more statistical methods. For example, secure protocols proposed for Bivariate Analysis [46], Correlation and Chi-Square Test [48] and Linear Regression [47] can be adapted to the proposed protocol.

3.3 Complexity

The time required by the proposed method to perform a secure multiparty computation task mainly depends on the number of participating organizations and the secure method in which they aim to compute securely. The proposed secure methods have different numbers of variables. So, if we denote the number of variables with α and the number of participating organizations with n, the computation cost for the proposed algorithm will be:

$$\text{Computation cost} = \alpha * n$$

These computations will be performed by a smart contract that uses the homomorphic addition property of Paillier cryptosystem to aggregate the partial variables, which has a complexity of $O(n)$ where n is the number of encrypted messages.

In addition, before the aggregation of the partial variables, the parties need to set up their own Paillier cryptosystem, encrypt their values and submit them to the blockchain network. However, these local computations could be performed once by each organization and in parallel. Therefore, they are negligible compared with the other computational costs, and we have not considered them. This also applies to the generation of the Paillier public key and decryption of the final value by the data user.

Finally, this complexity is in addition to the complexity added by the consensus protocol of the blockchain network, which differs based on the type of the blockchain and its study is beyond the purpose of this paper.

3.4 Method Comparison

Following, we discuss some of the important features of the proposed method compared to the related works. A summary of the comparison is provided in table 3.4.1.

1. **Auditability:** The consensus protocol in a blockchain network, ensures that only authorized transactions are committed to the distributed ledger. Transactions are logged and timestamped and stored on the ledger in forms of blocks.

So, the records are transparent and traceable. Therefore, any malicious activity, data manipulation, and illegal transactions will be easily detected.

2. **Flexibility:** We have implemented our method on Hyperledger Fabric, which is supported by giant tech companies such as IBM and SAP. The open-source and cross-platform features of Hyperledger Fabric provide data providers higher flexibility for adopting this solution.

3. **Availability:** As data are redundant between the peer nodes in a blockchain network, there is not a single point of failure in such networks, and users enjoy a higher data availability.

4. **Identity Management:** Hyperledger Fabric provides native APIs to control access to the resources. The access to resources can be regulated using Access Control Lists (ACL) or in more sophisticated cases using scripts. The peer nodes enforce the access control rules and ensure that only authorized users have access to the resources.

5. **Data Privacy:** We use the Paillier cryptosystem to encrypt confidential messages. In case an attacker gets access to the data stored on the blockchain, the attacker will not be able to read any of the query results as they are encrypted with the private key of the data user.

6. **Independent:** The proposed method does not rely on the addition of any agents or central servers for off-chain computations. All the calculations are done on-chain.

Features	[21]	[25]	[34]	[20]	Proposed Method
Availability	N	N	Y	Y	Y
Decentralized	N	N	N	Y	Y
Identity Management	N	Y	Y	Y	Y
Data Immutability	N	N	N	Y	Y
Data Auditability	N	N	N	Y	Y
Flexibility	Y	N	Y	Y	Y
Protected Private Key	Y	Y	Y	N	Y
Independent	Y	Y	Y	N	Y
Targeted Usage	Y	Y	Y	N	Y

Table 3.4.1: Comparison of the Proposed Method with Similar Works

3.5 Implementation

Several tools have been used during our implementations which will be discussed in this section.

3.5.0.1 Hyperledger Fabric

Hyperledger Fabric is an open-source permissioned blockchain platform developed by IBM and Linux Foundation. We used this platform to create our blockchain network.

3.5.0.2 Hyperledger Composer

Hyperledger Composer is an open-source framework to design, test and deploy business models for blockchain applications on Hyperledger Fabric. We used this framework to design our blockchain-based business model.

Figure 3.5.1 illustrates the structure of designed business network for the proposed method.

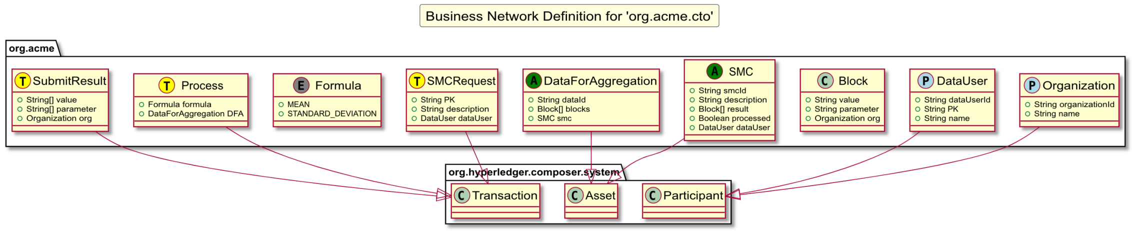

Fig. 3.5.1: Business Network Definitions

3.5.0.3 Composer REST Server

In order to communicate with the blockchain network on HTTP protocol, Hyperledger Composer REST Server [6] was used to generate a REST API from the deployed blockchain business network.

Figure 3.5.2 shows the running REST Server for our proposed model.

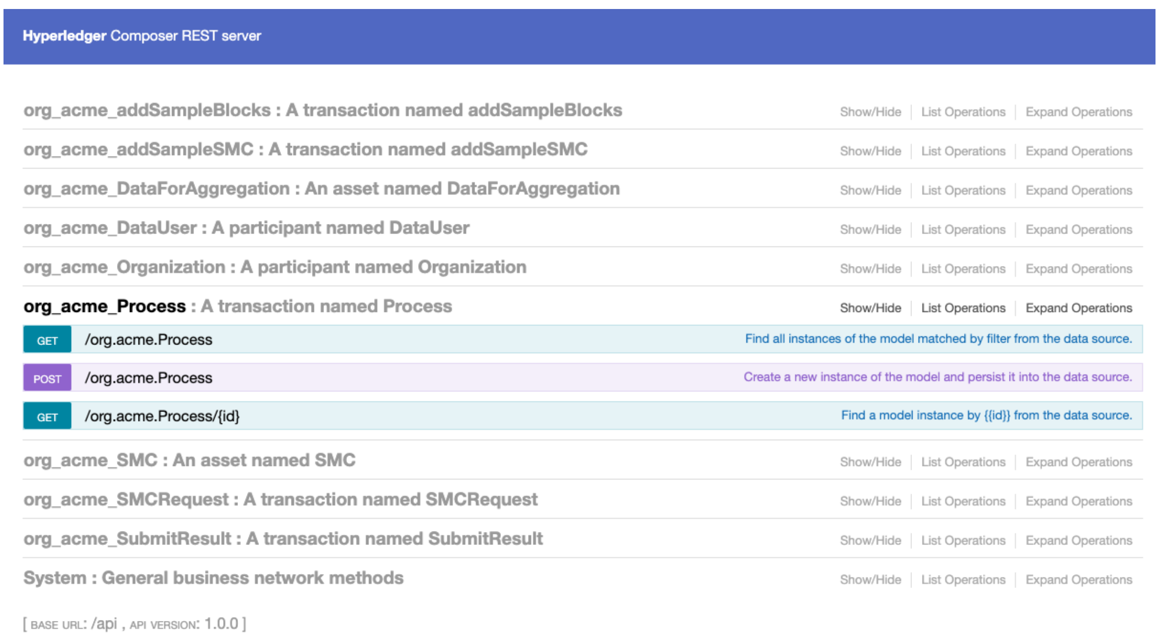

Fig. 3.5.2: Hyperledger Composer REST Server

CHAPTER 4

Experiments and Results

We evaluated the performance of our proposed method to show its viability in a distributed environment. The performance is evaluated by studying the latency of secure multiparty computations in various configurations.

4.1 Experiment Variables

There are two classes of variables in every experiment, dependent variables, and independent variables. The values of dependent variables depend on the values of independent variables. Response time of servers (RT) is usually considered as the bottleneck of blockchain applications, so we used this variable as the dependent variable. This variable describes the average amount of time that it takes for clients to receive a response after sending their requests to the servers.

4.1.1 Independent Variables

We have identified the following independent variables in our proposed method:

- **Key Size:** Key size in cryptosystems like Paillier is one of the important factors that determine how fast that cryptosystem runs. There is a trade-off between the key size and how secure the key is. Smaller keys run faster, but they are more prone to brute-force attacks.

- **Statistical Analysis Method:** Statistical analysis methods in our proposed scheme have different numbers of variables. For example, to compute the Mean method, two variables are to be computed by the parties. But for Variance, there are three variables. More variables result in more computations for the blockchain nodes.

- **Request Type:** There are two types of requests in a Hyperledger Fabric API, GET requests and POST requests. GET requests refer to requests that read data from the distributed ledger. These kinds of requests do not commit any changes on the ledger and therefore run faster. Conversely, POST requests are used for submitting transactions that modify the ledger. POST requests require reaching consensus among the blockchain nodes, and therefore they take a longer time to process.

- **Number of Organizations:** Our proposed algorithm works in a distributed environment, and each organization shares its part of computations. Increasing the number of organizations means more calculations and is expected to increase the overall execution time.

4.2 System Configuration

During the experiments, we used the following described system as the server (Table 4.2.1).

Operating System	Mac OS Cataline v10.15 Beta
Computer Model	MacBook Pro (13-inch, 2017)
Processor	3.5 GHz Intel Core i7
Memory	16 GB 2133 MHz LPDDR3
Container Platform	Docker

Table 4.2.1: System Specifications

4.3 Experiments

In the following experiments, we use each one of the abovementioned cases as the independent variable and consider the response time of the blockchain network as the dependent variable. We communicate with the blockchain network using a REST API provided by Hyperledger Composer. The REST API connects to a Hyperledger Fabric blockchain network and communicates with the clients using HTTP requests.

4.3.1 Key Size

We used four different key sizes for Paillier Cryptosystem to study its impact on our computations. Keys were generated using Paillier-js NPM Package [7] written in JavaScript language. The keys that were used are 512, 1024, 2048 and 4096 bits and are provided in tables 5.1.1, 5.1.2, 5.1.3, 5.1.4, respectively. NIST recommends 2048-bit keys as the standard key size. Therefore, we use a 2048-bit key (Table 5.1.3) in the next experiments. We used 10 organizations to jointly and securely compute the mean value of their datasets. Organization variables are summarized in table 4.3.1.

Organization	**A**	**N**
Organization 1	1000	250
Organization 2	2000	500
Organization 3	3000	750
Organization 4	4000	1000
Organization 5	5000	1250
Organization 6	6000	1500
Organization 7	7000	1750
Organization 8	8000	2000
Organization 9	9000	2250
Organization 10	10000	2500

Table 4.3.1: Mean Variables Used for Key Size Experiments

Table 4.3.2 shows the results. A summary of the results is provided in table 4.3.3 and illustrated in figures 4.3.2 and 4.3.1.

Table 4.3.2: Blockchain Response Time based on Size of Key

Request Number	512-Bit Key	1024-Bit Key	2048-Bit Key	4096-Bit Key
1	2.74s	2.88s	3.24s	4.38s
2	2.79s	2.99s	3.26s	4.57s
3	2.72s	2.89s	3.39s	4.44s
4	2.79s	2.85s	3.21s	4.38s
5	2.73s	2.83s	3.23s	4.42s
6	2.71s	2.85s	3.31s	4.27s
7	2.79s	2.81s	3.13s	4.38s
8	2.81s	2.82s	3.25s	4.21s
9	2.67s	2.80s	3.28s	4.19s
10	2.83s	2.82s	3.06s	4.38s

Key Size	Average RT	Standard Deviation
512	2.758s	0.0512
1024	2.854s	0.0560
2048	3.236s	0.0912
4096	4.362s	0.1131

Table 4.3.3: Response Time Results of Various Key Sizes

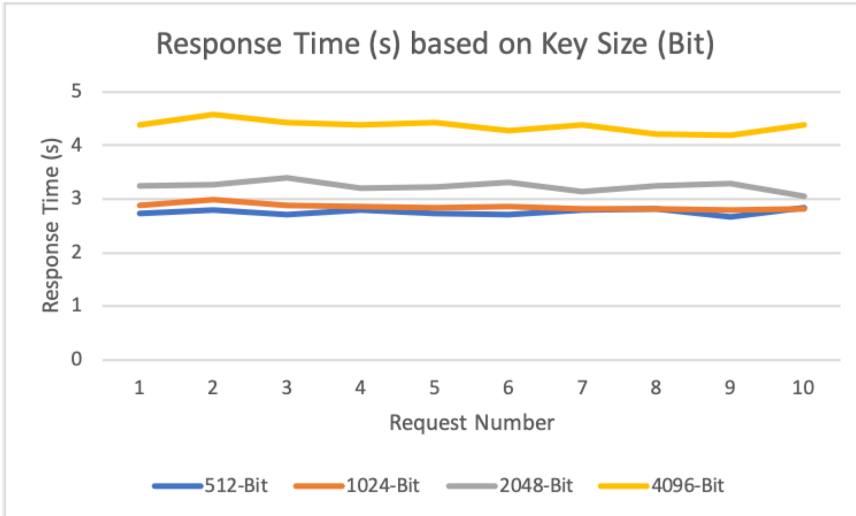

Fig. 4.3.1: Response Time (s) based on Key Size (Bit)

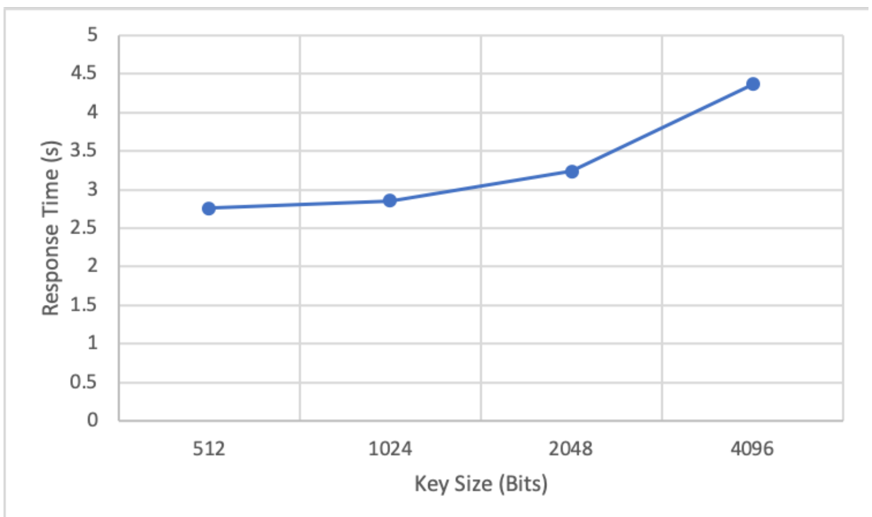

Fig. 4.3.2: Average Response Time based on Key Size

4.3.2 Statistical Analysis Method

Statistical functions have different numbers of variables in our proposed method.

The purpose of this experiment is to find the impact of increasing variables on the performance. We compare the proposed secure count, mean, and variance methods that have one, two, and three variables, respectively.

A summary of the variables used in this experiment is provided in table 4.3.4.

Organization	**Variables**		
	A	**B**	**N**
1	4	8	2
2	8	16	4
3	12	24	6
4	16	32	8
5	20	40	10
6	24	48	12
7	28	56	14
8	32	64	16
9	36	72	18
10	40	80	20

Table 4.3.4: Variables used for SMC

For each number of variables, we sent 10 requests to the blockchain network. The results are shown in table 4.3.5 and illustrated in figure 4.3.3 and figure 4.3.4.

Request	1-Variable	2-Variables	3-Variables
1	2.74	3.08	3.04
2	2.59	2.95	3.23
3	2.57	2.90	3.09
4	2.53	2.93	3.16
5	2.71	2.91	3.14
6	2.81	2.99	3.21
7	2.60	2.97	3.29
8	2.75	2.99	3.20
9	2.56	3.12	3.19
10	2.60	2.78	3.43

Table 4.3.5: Response Time (s) based on Number of Variables

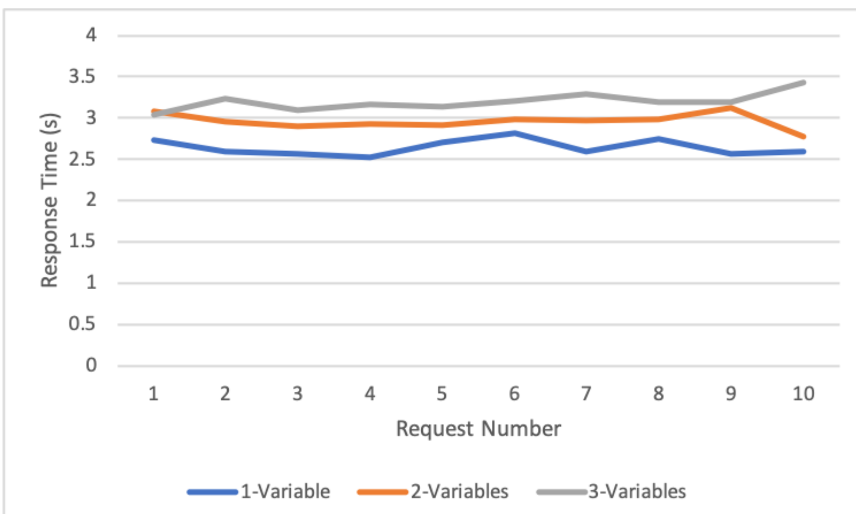

Fig. 4.3.3: Response Time based on Number of Variables

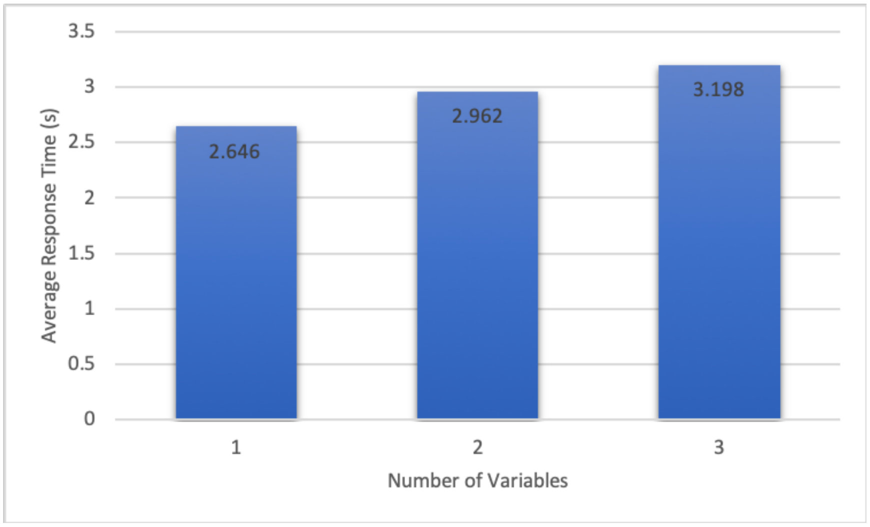

Fig. 4.3.4: Average Response Time based on Number of Variables

4.3.3 Request Type

Two types of HTTP requests, GET and POST, were used to communicate with the blockchain network. GET requests retrieve data from the blockchain, and POST requests submit transactions to the blockchain network. Nodes in a Hyperledger Fabric network store the latest value of variables in a separate database called State Database. GET requests are expected to operate faster since they make peer nodes to retrieve data from their local state database. Thus, we assume that the number of nodes does not have any impact on the latency for GET requests. Opposite to GET requests, POST requests are used to submit transactions to change the state of variables in blockchain, and these transactions need to go through a consensus process among the peer nodes, which take a longer time to execute [10].

In this experiment, we aim to find the time difference between regular GET and POST requests in the deployed blockchain network. A summary of the experiment variables and their values is shown in Table 4.3.6.

Independent Variables	**Dependent Variable**
Request Type	Response Time
GET, POST	

Table 4.3.6: Request Type Experiment Variables Summary

The results of this experiment is shown in table 4.3.7, illustrated in figure

Request	**GET**	**POST**
1	0.234	2.43
2	0.203	2.41
3	0.189	2.27
4	0.201	2.69
5	0.278	2.30
6	0.205	2.36
7	0.218	2.29
8	0.197	2.37
9	0.243	2.29
10	0.165	2.34

Table 4.3.7: Response Time (s) based on Request Type

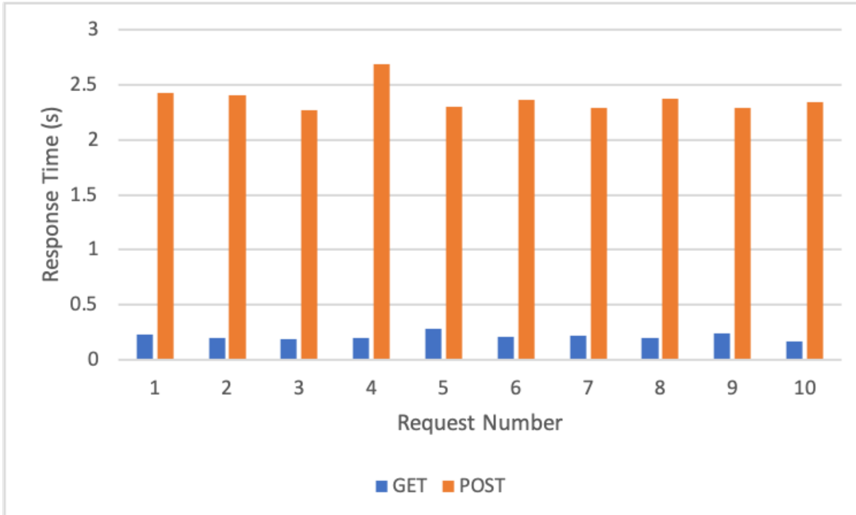

Fig. 4.3.5: Response Time based on Request Type

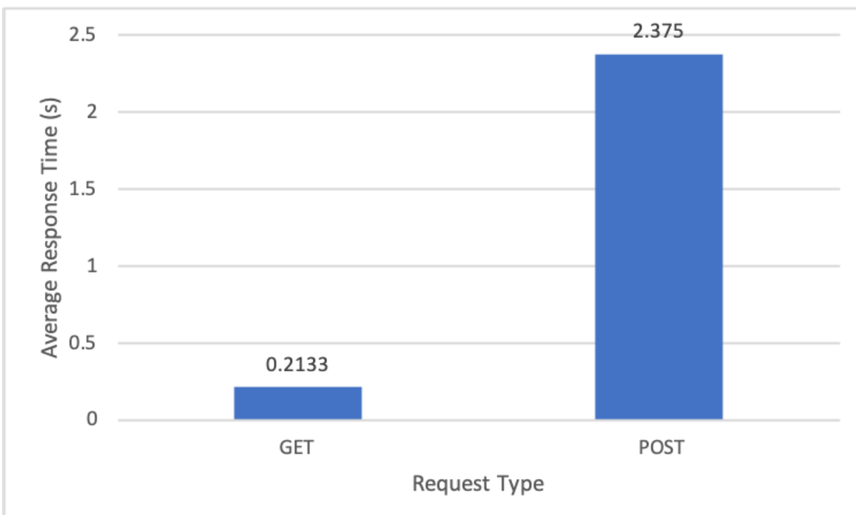

Fig. 4.3.6: Average Response Time based on Request Type

4.3.4 Number of Organizations

The number of participating organizations in a secure multiparty computation is another important variable that impacts the number of required calculations.

In this experiment, we used our model with different numbers of participating organizations. The variables for this experiment are summarized in table 4.3.8.

Independent Variables	**Dependent Variable**
Number of Organizations	Response Time
10, 20, 30, 40, 50	

Table 4.3.8: Request Type Experiment Variables Summary

The results for this experiment are shown in table 4.3.9, and illustrated in figure 4.3.7 and figure 4.3.8.

Organizations	**10**	**20**	**30**	**40**	**50**
Request 1	3.11	3.72	4.74	5.48	5.99
Request 2	3.06	3.67	5.4	5	5.59
Request 3	2.98	3.77	4.4	5.22	5.77
Request 4	4.03	3.76	4.41	5.29	5.79
Request 5	3.06	3.69	4.65	5.19	5.76
Request 6	3.02	3.7	4.41	5.08	5.89
Request 7	3.02	3.9	4.4	5.07	7.14
Request 8	3.04	3.75	3.31	5.18	6.06
Request 9	3.14	3.86	4.22	5.38	6
Request 10	3.2	4.8	4.46	5.05	7.27

Table 4.3.9: Response Time (s) based on Number of Organizations

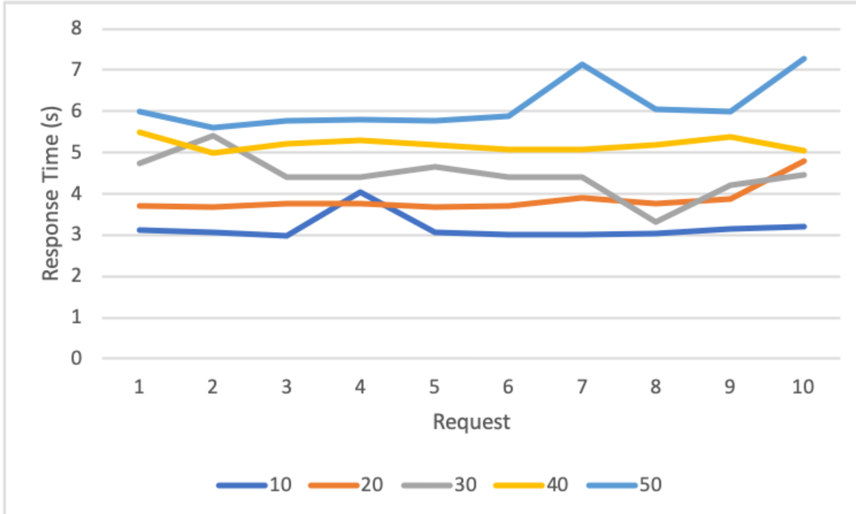

Fig. 4.3.7: Response Time based on Number of Organizations

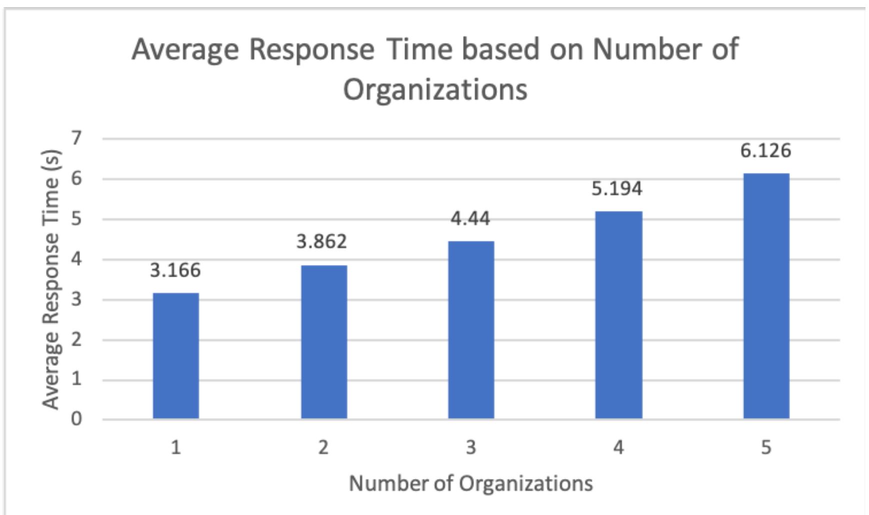

Fig. 4.3.8: Average Response Time based on Number of Organizations

CHAPTER 5

Conclusion

In this paper, we discussed the important usages of electronic health data, from education and regulation to public health. We examined the major obstacles to efficiently use these important data, which roots in strict privacy acts and isolated data silos. In this work, we proposed a novel solution for performing statistical analysis on private health data. We aimed to increase the accuracy of data analysis protocols while preserving the privacy of patients. To achieve this goal, we leveraged the blockchain technology and the Paillier encryption algorithm. Smart contracts were used to carry out mathematical operations on the encrypted records in a secure manner. We were able to successfully deploy the proposed scheme on the Hyperledger Fabric permissioned and consortium blockchain platform. Our experimental results showed the feasibility of this method with an average of 3 seconds processing time for 10 organizations that securely compute a regular mean statistical method. We also tested our method with both the standard 2048 key size and a larger 4098 key size. The 4098 key size increased the response time by 1.2 seconds. However, this overhead highly depends on the computation power of the blockchain nodes and is expected to decrease as computers become more powerful.

5.1 Future Work

Our method was tested on a single blockchain node. To precisely examine the scalability of this method, a larger environment would be preferable. As part of our future work, we aim to test the scalability of the proposed method in a large network of blockchain nodes.

Another area for efficiency improvement is detaching the external libraries from the smart contracts. When data are encrypted using a standard 2048 key size in a Paillier cryptosystem, they would have a length of around 1233 digits; however, the JavaScript language, which is one of the languages for writing smart contracts in Hyperledger Fabric, does not have native support for such big integers. Consequently, we needed to add the Big-integer JavaScript library [8], which adds big integers support to the language, to our smart contracts. As the results of our experiments show, the main bottleneck of our proposed method is the data aggregation part, which is carried out by the smart contracts. Therefore, we expect that by detaching the Big-integer library from our smart contracts and by using a native approach, we can increase the computation speed and achieve a better result. The development such a native approach is another part of our future works.

Moreover, We adapted four statistical methods, which are count, mean, variance, and skewness, in our proposed framework to demonstrate its feasibility. Adaptation of more secure statistical methods is moreover in our future plans to improve this research.

Furthermore, the proposed method only supports simple arithmetic addition operator, through Paillier cryptosystem, that limits its ability to calculate more sophisticated queries that require either multiplication or comparison of values. Algorithms like ElGamal [26] and Goldwasser-Micali (GM) [30] can be adapted to add arithmetic multiplication and XOR operators to overcome this limitation.

Lastly, scalability is a known problem of blockchain applications [35] that is still an open problem, and under research by the time of writing this thesis. This limitation also applies to our proposed protocol, and solutions for further improvements of the scalability is another future plan of this thesis.

APPENDICES

This chapter contains additional information, tables, and figures.

Table 5.1.1: 512-Bit Paillier Cryptosystem Key

Public Key

Variable	**Value**
n	93655871196023231196795485505548417030107063052369830300460 70791890157263254383105502324241421169259451283255449738760 1431248796674159300349028711994473 03
g	53837341849265314450281255661502354314937180675206978044254 17571285052197101181170721871388578743809992621735097517359 38390099185558879335039659911275764336491053323802510787812 25728911685337990307293660644724939329124693620816711982023 44479386738739558705812418549766806616538148703326663213219 8862053023810

Private Key

Variable	**Value**
p	97071756839829766151357712311346040343611102328927696045539 071769137169775591
q	96481071575285475834754052683174351953573318911494146224074 280476635870180833
mu	74311348010141642413144988458723338964000012170380501177185 79022675181279042648559622591695316375485493931713436642111 362968264564738309155775786677065813

lambda 4682793559801161559839774275277420851505353152618491515023035395945078631627094776336954563089591573843144367528720787860942228912573158341328549079745440

Table 5.1.2: 1024-Bit Paillier Cryptosystem Key

Public Key

Variable	**Value**
n	92186359388720763500220923692654526314137689500074317157520 54447744303374942883452863073637924123283709953024371273233 61194231002452161999125557924555532924469075037672285115775 38564306243309621340482063268130972845277502795232734873572 42114880219124056922040978947243525789670435974095189093120 8922734124813
g	81247765265401369308721959597962425152354744136073810613824 12413918650243615699659674453288037142179826298841496400056 21826050884690380431611312742182833844330087358604348147873 21225416054156812482586915314685021976523997139374895215612 75927425188333360230017543680974536603182605603839876097073 72824176353593568008060932358504884837854674135577403919034 51074635883192012930195040164306417653864873640551619890365 38635889943462184368644930627067374132658338543989054003763 59318165633262227939545279131919409184966677196450664483993 90172198895567015754806550979386427307320949264681708468942 90431433453250666507396921

Private Key

Variable	Value
p	9249845663918815382435707879401140075139024418123977866299154679697969677261012175730730374704090690967634612608959302816008495008154341714629823621497359
q	9966259193741491430079220509498931275769465297835518971680888461194624478234283278002024199967404503036617744268601253460515159006184003404814596448188707
mu	68218657810016434308207901543249782662480871927408939811462039028933173331059884780920383884799035774097678084980302734037948495702809649032030978991109719578379831422079122059387547380192924226011684711538645824482608458121472095507664168124579297609077278135254623804037287735239274327878256712235390781462
lambda	46093179694360381750110461846327263157068844750037158578760272238721516874714417264315368189620616418549765121856366168059711550122608099956277896227776636615401323053460849531305087703085979356425383051885646496401068305100538619789059344197113759872687608078716297437350810090352863306772905882251332219374

Table 5.1.3: 2048-Bit Paillier Cryptosystem Key

Public Key

Variable	**Value**
n	21713054676660844786103147515999649098796097557026994195757586 671569258087951532626009971575152790369115283258370504843684893 589401288976156158948654454698803175233942055117632444076130802 769071815971431934329685176828915561687860101549931964408248243 602122473739216853948603926664206443061479280622852970382740704 021755687542320753779461859030600254490293539917665027894225793 176154303340174600028204926868732470963732627563991372791499531 684825919797445747278230636171315334286085033539132484666156999 161067201547105981764741062495971470367455633213087934936648018 76904576422355526249646204799452416448286073412343 9

g	26848920958653350475896021303073200168866628430034994322073146 11901905593247085635327000239706247841105858749190762274830951 5 25190550958647269904455146938252313561640031663891090558422436 5 89334836143095613866479262232309162702006435875670740952618435 7 77810785637258959530265792624226269854949953872449374471785173 5 17587966314833455550729527642220535666614497910305686788765137 5 15148911043660573205415309790584181469303498985971701990282650 3 76574656246862701454287430758401066644970596069490244595397877 2 75174354249223451411366892313541338883132021212740752984391111 6 22723442557632082838622295869574897215307823008952640051247018 6 51753232800271725020877291170974449461273052621386340645846102 6 44306986412321897408575669630954423036072509593170515704297634 8 93994334675257642171862970186580280073622041567803998050492004 0 71902327776603120472784351895051436587758178550889055598541260 7 86735380959994301308388612427352273295593687408333742281582765 1 20614257308092712052479649492756677790125491988613754311460801 5 02669021091891317395672298192280677757593108754697133151002898 9 64432073545391382849539304886758665159678132842914895238354486 2 42773952993561009843789312274122013719954034348004816511650341 5 219363157192083057017143133673385602

Private Key

Variable	**Value**
p	137440786767258822296245082664583461829343127408488373023822563 607668476602841452055542059474497191407374626511489818763361939 321913225492634510064955178864438501971536401055550661673657631 913629041211505728299087603751019260424322554267661338215361962 6240421104662482365904087980384488146826566131646582271 61

q 1579811581945435571414796024751279355460975511668778363467667642856839148594704381430179012772648704014129718148346060350037043983370207026622996783545533611529225497468365405837647704015743055866869458972859982036619028590468010185048000355925610910766099228818733044780794733324682693832294345457882278145990

mu 1584069366844111390574676161113641255321528591225791083061817198503601302298978736557783063461265051529802707520633156775761494805781865756224195349121448326192605212792400923259885867301463094930913004176267623655618835532675839449468270777300353442825133136758370523812928908138001177443523853257169965783840163975031360317368228420827083884484838486759637718561561237963683449294957292086441440694405356698306916747357653669059768549278903848810407334165305903560218923905843947180817822204412505118891469424708223066517801728238762861812371541997247826024533594965256236474080402006492663154421158751004477420488

lambda 1085652733833042239305157375799982454939804877851349709787879333578462904397576631300498578757639518455764162918525242184244679470064448807807947432722734940158761697102755881622203806540138453590798571596716484258841445778084393005077496598220412412180106123686960842697430196333210322153073964031142648519122264103839694258144151454716107381661240690585908685414721928316622039234599572498802003372658240346184168270315156959650357388964071931531149385121877300252237339702552424444955634421272996889171645489926640434524954359209437306465419396814115574094870702688835310575444269018297766489830603666195392404084

Table 5.1.4: 4096-Bit Paillier Cryptosystem Key

Public Key

Variable	**Value**
n	5372474845836635807163958394362306230164352156572954390892032403026799093664093656764570147750641775777399538221769925662072033860265944889967097970133488781033261227629411723459139152196220813004717969150866956885005670232632231576610809638561609057960355506647088305179480510500018238128374068629239506487405624559145390682750072276130988538342369626277263992328460831689153462973395853999520044872495261630191548934716218055833153471014238585758906794522965112175491805091607477267928978367406468151675817340843574161166684681477643277722889653657867399813891919844317909262436668412663094269972714192708310769841517216886128580119720247829549066363927933595043936628705627276237605872144195259625697945312570900094102108168565775914526932195105009545865256194142565366040470510979311108341778199127789350675593095514069531180337497276653638538496755364865664105777015123223957725805545713050579255453957687046539431208979701133006884587626142689048885880384781804773246979811308001182964661174564309579787163881482844442306897970672374769689471521281923184463643769322752463461618727098948108290132655496993131794365472934577945250269182081083051639222235829726682061160028006485332845522909781668096685867783087556120320586943679

g	537247484583663580716395839436230623016435215657295439089203240 302679909366409365676457014775064177577739953822176992566207203 386026594488996709797013348878103326122762941172345913915219622 081300471796915086695688500567023263223157661080963856160905796 035550664708830517948051050001823812837406862923950648740562455 914539068275007227613098853834236962627726399232846083168915346 297339585399952004487249526163019154893471621805583315347101423 858575890679452296511217549180509160747726792897836740646815167 581734084357416116668468147764327772288965365786739981389191984 431790926243666841266309426997271419270831076984151721688612858 011972024782954906636392793359504393662870562727623760587214419 525962569794531257090009410210816856577591452693219510500954586 525619414256536604047051097931110834177819912778935067559309551 406953118033749727665363853849675536486566410577701512322395772 580554571305057925545395768704653943120897970113300688458762614 268904888588038478180477324697981130800118296466117456430957978 716388148284444230689797067237476968947152128192318446364376932 275246346161872709894810829013265549699313179436547293457794525 026918208108305163922223582972668206116002800648533284552290978 1668096685867783087556120320058694368

Private Key

Variable	**Value**
p	2497408952068069972666274366874391074894999368251257801120030813419368285189494200418205924905753806401257435048591710447136921393478623086997780671078727660519187704182286972166581524929312784629439352429442556690359778527228519268493683193195862919805641813485655044414335476797497292910238848187919397006204587733137065368810586759373682130010724872841352038254843620658985166631758495291636030772750414937945355983229683528173234826102763958112947582832679874961239663651712087505620085637590421432198770051780846988187604271787944264254809865254653676361716770926738246655110857873853888205529439203809206843 2051
q	2151219503472894716509129472987625051882207136607845989354155228529961552222586633431027798907459894273033661732624909948404064449212439835862202264049577732271669018949994050050186629464098271409934384385036044072071373606851073433061955509771191753233465819740227793186549945100545733566072331185335276016577073578045149640274230816815511835533225396475875662015093990461194209584000133172471926220422471596896145095671115156837245361750862585325983701145397630490415461243710665909651932601155810263621372732366666932010926397293848153643943404560408669077585154243072831068719068447701295268606620581961935512 4517

mu 4879168444868919951703702574577320193578745911395894698946366802565885816989970247124329847363525723217461936097341535431830081306540589508457542153429501026632090102871916422990846400899935076213190178948921221267165484448561662538217828023924677405202694389930874488493656167704678656717840314618958850202248454504527184310570172580612572631878467877091083870177602353742730515906669374259625797078815278716033645274532768630097723953111674997868249326866663943473005164444248694593554289192473580168247474902995844447678995548304749437938380988531379103821222561697644766898920074744102472550448707310267586477360597366247132588359754344694238816106988091836494276524044327383673583129813963517980986811817811502272369227385753906494825783082809125146039857212799488927730983640878095258686530597942277881372563390614550944604875468975893217511485181336814779337467740926004402646318962686160650835493876789017880428800432090967102824437655373663737732442171369370980957807379503883509829984833994975868453035481156178198168803930900134659900613828448143212339054404313468889012247987241647176939716509162288907299025616425991790719472558094090401444446112871642303921991465856482330988576227800758163221708405758617240355899931194

lambda 537247484583663580716395839436230623016435215657295439089203240302679909366409365676457014775064177577739953822176992566207203386026594488996709797013348878103326122762941172345913915219622081300471796915086695688500567023263223157661080963856160905796035550664708830517948051050001823812837406862923950648740562455914539068275007227613098853834236962627726399232846083168915346297339585399952004487249526163019154893471621805583315347101423858575890679452296511217549180509160747726792897836740646815167581734084357416116668468147764327772288965365786739981389191984431790926243666841266309426997271419270831076984105235404057448365080270744556286475125021294455802624965820867204267288840298717624077457293124953002667299849044411387497283361083590325357925790062973482676138483866608300612010138368844824507165572183262167110409438206324567926834119149456895863847310435990136944204545717086077495280632656910921396390670153500188866308671766093142996648383038677784152420978431424007094672355298844673337636818216555578882274786278229490118842350249656053011975064092494471291645321460940583294860545367311850862478345865616319385824932901417487239743236050274517582661609781396835173775052678903594116752036948157754320635137800

REFERENCES

[1] (2001-2019). Bittorrent, inc. bittorrent web site. `bittorrent.com`.

[2] (2003). *Key Capabilities of an Electronic Health Record System: Letter Report*. The National Academies Press, Washington, DC.

[3] (2004). Personal health information protection act. `https://www.ontario.ca/laws/statute/04p03`.

[4] (2018). Blockchain and healthcare: the estonian experience. `https://e-estonia.com/blockchain-healthcare-estonian-experience/`.

[5] (2018). Hyperledger fabric now supports ethereum. `https://www.hyperledger.org/blog/2018/10/26/hyperledger-fabric-now-supports-ethereum`.

[6] (2018). Ledger - hyperledger composer rest server. `https://hyperledger.github.io/composer/v0.19/reference/rest-server`.

[7] (2018). Paillier-js: A node.js implementation of the paillier cryptosystem. `https://www.npmjs.com/package/paillier-js`.

[8] (2019). Biginteger.js - an arbitrary length integer library for javascript. `https://www.npmjs.com/package/big-integer`.

[9] (2019). Hyperledger composer modeling language. `https://hyperledger.github.io/composer/v0.16/reference/cto_language.html`.

[10] (2019). Ledger - hyperledger fabric documentation. `https://hyperledger-fabric.readthedocs.io/en/release-1.4/ledger.html`.

[11] (2019). Why the next evolution of global health care will be blockchain-based. `www.forbes.com/sites/chrissamcfarlane/2019/02/07/why-the-next-evolution-of-global-health-care-will-be-blockchain-based/`.

[12] Abouzahra, M. (2019). Using blockchain technology to enhance the use of personal health records.

[13] Act, A. (1996). Health insurance portability and accountability act of 1996. *Public law*, 104:191.

[14] Alonso, S. G., Arambarri, J., López-Coronado, M., and de la Torre Díez, I. (2019). Proposing new blockchain challenges in ehealth. *Journal of medical systems*, 43(3):64.

[15] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al. (2018). Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM.

[16] Assistance, H. C. (2003). Summary of the hipaa privacy rule. *Office for Civil Rights*.

[17] Azaria, A., Ekblaw, A., Vieira, T., and Lippman, A. (2016). Medrec: Using blockchain for medical data access and permission management. In *2016 2nd International Conference on Open and Big Data (OBD)*, pages 25–30. IEEE.

[18] Barker, E., Burr, W., Jones, A., Polk, T., Rose, S., Smid, M., and Dang, Q. (2009). Recommendation for key management part 3: Application-specific key management guidance. *NIST special publication*, 800:57.

[19] Bayardo, R. J. and Agrawal, R. (2005). Data privacy through optimal k-anonymization. In *21st International conference on data engineering (ICDE'05)*, pages 217–228. IEEE.

[20] Benhamouda, F., Halevi, S., and Halevi, T. T. (2019). Supporting private data on hyperledger fabric with secure multiparty computation. *IBM Journal of Research and Development*.

[21] Boroujerdi, A. S. (2016). *Privacy-preserving statistical analysis methods and their applications on health research*. PhD thesis, Memorial University of Newfoundland.

[22] Castro, M., Liskov, B., et al. (1999). Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186.

[23] Chen, Y., Ding, S., Xu, Z., Zheng, H., and Yang, S. (2019). Blockchain-based medical records secure storage and medical service framework. *Journal of medical systems*, 43(1):5.

[24] Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., and Zhu, M. Y. (2002). Tools for privacy preserving distributed data mining. *ACM Sigkdd Explorations Newsletter*, 4(2):28–34.

[25] Drosatos, G. and Efraimidis, P. S. (2011). Privacy-preserving statistical analysis on ubiquitous health data. In *International Conference on Trust, Privacy and Security in Digital Business*, pages 24–36. Springer.

[26] ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472.

[27] Firdaus, A., Ab Razak, M. F., Feizollah, A., Hashem, I. A. T., Hazim, M., and Anuar, N. B. (2019). The rise of "blockchain": bibliometric analysis of blockchain study. *Scientometrics*, pages 1–43.

[28] Goldreich, O. (1998). Secure multi-party computation. *Manuscript. Preliminary version*, 78.

[29] Goldreich, O., Micali, S., and Wigderson, A. (1987). How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229. ACM.

[30] Goldwasser, S. and Micali, S. (1982). Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 365–377. ACM.

[31] Khezr, S., Moniruzzaman, M., Yassine, A., and Benlamri, R. (2019). Blockchain technology in healthcare: A comprehensive review and directions for future research. *Applied Sciences*, 9(9):1736.

[32] Liew, C. K., Choi, U. J., and Liew, C. J. (1985). A data distortion by probability distribution. *ACM Transactions on Database Systems (TODS)*, 10(3):395–411.

[33] Machanavajjhala, A., Gehrke, J., Kifer, D., and Venkitasubramaniam, M. (2006). l-diversity: Privacy beyond k-anonymity. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 24–24. IEEE.

[34] Marwan, M., Kartit, A., and Ouahmane, H. (2016). Applying secure multi-party computation to improve collaboration in healthcare cloud. In *2016 Third International Conference on Systems of Collaboration (SysCo)*, pages 1–6. IEEE.

[35] McGhin, T., Choo, K.-K. R., Liu, C. Z., and He, D. (2019). Blockchain in healthcare applications: Research challenges and opportunities. *Journal of Network and Computer Applications*.

[36] Mettler, M. (2016). Blockchain technology in healthcare: The revolution starts here. In *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pages 1–3. IEEE.

[37] Mikula, T. and Jacobsen, R. H. (2018). Identity and access management with blockchain in electronic healthcare records. In *2018 21st Euromicro Conference on Digital System Design (DSD)*, pages 699–706. IEEE.

[38] Miller, V. S. (1985). Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques*, pages 417–426. Springer.

[39] Nakamoto, S. et al. (2008). Bitcoin: A peer-to-peer electronic cash system.

[40] Nchinda, N., Cameron, A., Retzepi, K., and Lippman, A. (2019). Medrec: A network for personal information distribution. In *2019 International Conference on Computing, Networking and Communications (ICNC)*, pages 637–641. IEEE.

[41] Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer.

[42] Reiss, S. P. (1984). Practical data-swapping: The first steps. *ACM Transactions on Database Systems (TODS)*, 9(1):20–37.

[43] Rivest, R. (1992). The md5 message-digest algorithm.

[44] Rivest, R. L., Adleman, L., Dertouzos, M. L., et al. (1978a). On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180.

[45] Rivest, R. L., Shamir, A., and Adleman, L. (1978b). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.

[46] Samet, S. (1988). Secure bivariate statistical analysis on health data. *New Engl. J. Med*, 318:262–264.

[47] Samet, S. (2015). Privacy-preserving logistic regression.

[48] Samet, S., Boroujerdi, A. S., and Asghari, S. (2016). Secure health statistical analysis methods. *IADIS International Journal on Computer Science & Information Systems*, 11(1).

[49] Sheikh, R., Mishra, D. K., and Kumar, B. (2011). Secure multiparty computation: From millionaires problem to anonymizer. *Information Security Journal: A Global Perspective*, 20(1):25–33.

[50] Siyal, A., Junejo, A., Zawish, M., Ahmed, K., Khalil, A., and Soursou, G. (2019). Applications of blockchain technology in medicine and healthcare: Challenges and future perspectives. *Cryptography*, 3(1):3.

[51] Stephanie Newkirchen, Natasha Elsner (2018). Electronic health records: Can the pain shift to value for physicians? `https://www2.deloitte.com/insights/us/en/industry/health-care/ehr-physicians-and-electronic-health-records-survey.html`. [Online; accessed 21-August-2019].

[52] Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570.

[53] The Economist Intelligence Unit (2017). Healthcare rallies for blockchains: Keeping patients at the center. `http://www.ibm.biz/blockchainhealth`. [Online; accessed 21-August-2019].

[54] Traub, J. F., Yemini, Y., and Woźniakowski, H. (1984). The statistical security of a statistical database. *ACM Transactions on Database Systems (TODS)*, 9(4):672–679.

[55] Wikipedia contributors (2019a). Proof of stake — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Proof_of_stake&oldid=911001824`. [Online; accessed 18-August-2019].

[56] Wikipedia contributors (2019b). Proof of work — Wikipedia, the free encyclopedia. [Online; accessed 18-August-2019].

[57] Wood, G. et al. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32.

[58] Yao, A. C.-C. (1982). Protocols for secure computations. In *FOCS*, volume 82, pages 160–164.

[59] Yao, A. C.-C. (1986). How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167. IEEE.

VITA AUCTORIS

NAME:	Mahdi Ghadamyari
PLACE OF BIRTH:	Mashhad, Iran
YEAR OF BIRTH:	1994
EDUCATION:	University of Bojnord, B.Sc. in Computer Science, Bojnord, Iran, 2016
	University of Windsor, M.Sc. in Computer Science, Windsor, Ontario, 2019