Security Stratagies and Blockchain Functionalities for Genomics Analytics in Bioinformatics

Abukari Mohammed Yakubu

Bachelor of Science in Computer Engineering

Master of Science in Applied Computer Science and Society

A thesis submitted in total fulfilment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science and Information Technology

School of Engineering and Mathematical Sciences

College of Science, Health and Engineering

La Trobe University

Victoria, Australia

November 2021

Table of Contents

List of Figures

List of Tables

List of Publications

1. Mohammed Yakubu, A., & Chen, Y.-P. P. (2020). Ensuring privacy and security of genomic data and functionalities. *Briefings in bioinformatics*, 21(2), 511-526.

2. Mohammed Yakubu, A., & Chen, Y. P. P. Privacy-based Drug Dosage Prediction in Genotype and Clinical Personalized Medicine. *ACM Transactions on Privacy and Security*. (Under Review).

3. Mohammed Yakubu, A., & Chen, Y. P. P. A Blockchain Implementation for Controlling Genomic Access and Variant Discovery Using Smart Contracts and Homomorphic Encryption. *Future Generation Computer Systems*. (Under Review).

4. Mohammed Yakubu, A., & Chen, Y. P. P. Blockchain Functionalities for Privacy and Integrity in Genomic Analyses. *ACM Computing Surveys*. (Under Review).

5. Mohammed Yakubu, A., & Chen, Y. P. P. Secure Federated Learning for DNA Sequence Classification with Verifiable Gradient Aggregation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. (Under Review).

Keywords

Genomics, Genomic security and privacy, Privacy attacks on genomic data, Privacy-preserving, Secure computation, Homomorphic encryption, Secure multiparty computation, Personalized medicine, Warfarin dosage, Blockchain, Smart contracts, Access control, Deep learning, Federated learning, Verifiable computation.

Abstract

Over the past decade, technological advances in DNA sequencing have resulted in the growth of genomic biobanks and repositories. These genomic data are used by researchers to advance biomedical research in areas of personalized medicine, diagnostic testing, and drug discovery. However, performing analytical tasks on genomic data faces the challenges of security, privacy, and access control for individuals whose DNA samples are contained in the genomic repositories. In this thesis, we propose novel security strategies and blockchain functionalities to address these challenges to (i) predict patients drug dosage in personalized medicine, (ii) integrate blockchain smart contracts to control genomic access, and (iii) federated deep learning for DNA sequence classification. First, we investigate current privacy attacks on genomic data and examine the cryptographic primitives that have been utilized to protect genomic privacy. Then we investigate current state-of-the-art genomic privacy-preserving techniques and provide a comprehensive analysis based on their encryption techniques, security goals and computation overheads. Second, we develop new secure techniques in personalized medicine to predict warfarin dosages specific to patients' genetic variants using warfarin regression models, homomorphic encryption and secure two-party computation protocols. To the best of our knowledge, our work is the first to develop a secure genotype-guided dosage prediction framework. Third, we propose a novel approach to integrate blockchain smart contracts for genomic access control and variant discovery. For our variant discovery protocol, we develop new query optimization techniques using genomic data partitions, binary search trees and bloom filters. Finally, we develop a new federated learning framework for DNA sequence classification to protect genomic and neural network model privacy. We propose a new technique to minimize the weight divergence between training parties to improve model accuracy which is superior to state-of-the-art techniques. Our evaluation results demonstrate that our proposed secure frameworks guarantee genomic privacy with efficient computation cost and communication throughput.

Statement of Original Authorship

Except where reference is made in the text of the thesis, this thesis contains no material published elsewhere or extracted in whole or in part from a thesis submitted for the award of any other degree or diploma. No other person's work has been used without due acknowledgment in the main text of the thesis. This thesis has not been submitted for the award of any degree or diploma in any other tertiary institution.

Abukari Mohammed Yakubu
2nd November 2021

Acknowledgments

First and foremost, I would like to express my sincere appreciation to my principal supervisor, Professor Yi-Ping Phoebe Chen, for her guidance, encouragement, and continued assistance from the commencement to the completion of my PhD studies. I would also like to thank her for accepting me as a PhD student under her supervision and mentoring me to grow as a research scientist. Her deep insights helped and directed me at every stage of my research. I would also like to acknowledge La Trobe University Postgraduate Research Scholarship and La Trobe University Full Fee Research Scholarship for supporting my research work and candidature.

I would also like to thank my mother and father fo all their love and support throughout these years. This accomplishment would not have been possible without them. Furthermore, a special thanks to my siblings for their continued encouragement during this amazing journey. Finally, all thanks to the almighty God for giving me life, blessings and seeing me through my PhD studies.

Chapter 1: Introduction

1.1 Background and Motivation

Recently, next-generation DNA sequencing technologies have been widely used by scientists to rapidly sequence entire genomes due to its low cost and high-throughput capacity. This has led us into the genomic big data era where most of these data are shared on public repositories and biobanks such as the UK biobank [1], the ENCODE project [2] and the 1000 Genomes Project [3] to aid clinicians and researchers to advance biomedical research into understanding the genomic structure and the functionalities of biological data (DNA, RNA and proteins). These advances have ushered in opportunities and benefits in several application domains such as (i) genetic and diagnostic testing: diagnosing an individual by matching certain disease biomarkers with the persons DNA sequence [4], (ii) personalized medicine: treatment and prescription medications are tailored for an individual based on his genetic profile [5], and (iii) genome-wide association studies (GWAS): studying and discovering variants and genetic regions which are responsible for certain diseases and traits [6].

The aforementioned benefits of genomics are confronted with the serious challenges of security, privacy, access control and confidentiality of the individuals whose DNA samples are contained in genomic data repositories and biobanks. This is because the human genome is prone to security and privacy attacks (i.e., identity tracing, attribute disclosure and completion attacks) as it is highly sensitive, correlates amongst family members and contains an individual's disease susceptibility markers [7]. In addition, genomic data custodians such as research facilities lack the storage and computing resources to analyse large volumes of genomic data so they outsource them to high-end third-party cloud environments (e.g., Google Genomics [8] and Microsoft Genomics [9]). This exacerbates the security situation as third-party cloud environments are semi-trusted and can infer sensitive information from the DNA sequences of individuals. Data privacy regulations including the health insurance portability and accountability act (HIPAA) and the general data protection regulation (GDPR) are required to enforce best security and privacy practices when dealing with genomic data. Ethics

and safety committees of hospitals and biomedical research facilities have policies in place outlining how genetic information should be used without compromising privacy. These policies include safeguards that must be implemented under HIPAA and GPDR privacy rules to prevent unauthorized use or disclosure of genomic data [10]. However, there are certain exceptions (e.g., required by law) which permits the disclosure of genetic information without the consent of individuals [11]. These exceptions are loopholes that results in genomic privacy breach cases. A widely used technique to safeguard data privacy that is included in HIPAA's and GPDR's privacy regulations is to employ data anonymization techniques to conceal or exclude personal identifying information (e.g., names, addresses etc.) that links individuals to their DNA data. Several studies have investigated the genomic privacy guarantee provided by data anonymization techniques [12, 13, 14]. These studies have identified that (i) HIPAA's and GPDR's privacy regulations do not address the genomic privacy issue from a cryptographic point of view, and (ii) data anonymization techniques are vulnerable to attribute disclosure attacks when the adversary has prior knowledge about the presence of an individual in a genomic dataset. In addition, these studies suggest that cryptographic and encryption techniques will provide more secure and improved measures for protecting genomic data as opposed to the anonymization techniques. A common cryptographic approach to secure genomic data is to encrypt the data with symmetric encryption techniques, for example the advanced encryption standard (AES). This, however, is limited as it does not allow computation and analyses on encrypted data. With this in mind, homomorphic cryptosystems which facilitate genomic analyses (e.g., genetic testing) to be performed on encrypted DNA sequences without prior decryption are of the utmost significance to guarantee genomic privacy, security and confidentiality.

This thesis explores how to protect the security, privacy and confidentiality of genomic data at storage and during genomic analytical computation with minimal overheads in computation and transmission throughput. Specifically, the goal of this thesis is to develop new cryptographic approaches to address the security vulnerabilities and protect genomic data while allowing genomic analyses on encrypted data using

(i) cryptographic building blocks [15]: public key encryption, homomorphic encryption, secure multiparty computation techniques, digital signatures and non-interactive zero knowledge proofs, and (ii) blockchain functionalities [16]: smart contracts, decentralization, non-repudiation, transparency, and immutability. In this thesis, we focus on proposing cryptographic techniques to solve security problems in genomic analyzes for (i) personalized medicine, (ii) controlling genomic access and variant discovery, (iii) blockchain functionalities to preserve genomic data privacy, and (iv) federated learning for DNA sequence classification. In addition, in this thesis we aim to find solutions which guarantee the following requirements.

- **Security:** No sensitive information about an individual's genomic data such as single nucleotide polymorphisms (SNPs) should be disclosed to unauthorized parties.
- **Privacy:** The right of an individual to conceal his personal identifying information from genomic repositories and datasets should be respected.
- **Access control:** Individuals should be able control access to their DNA sequences after contributing them to genomic repositories, and should be able to grant or deny access rights to other stakeholders.
- **Computational efficiency:** The runtime cost to process our proposed secure genomic analytics task (e.g., dosage computation) should be minimal and suitable for practical genomic applications. In addition, the computations should be superior and supported by third-party environments such as the cloud.
- **Transmission efficiency:** The cost to transmit encrypted DNA sequences between parties in our proposed secure frameworks should be minimal and practically feasible.

1.2 Research Problems

This thesis focuses on performing genomic analyses operations in encrypted domain. There are many cryptographic techniques and the selection of an encryption technique

to protect the privacy of genomic data is vital in determining the scalability and feasibility of the genomic processing application. To protect genomic privacy, specific encryption algorithms that suit the genomic application use-case and computational task should be used. For example (i) to perform linear operations on encrypted genomic data, homomorphic encryption is selected, (ii) for comparison operations and nonlinear operations, secure multiparty computation (SMC) is best suited, and (iii) for access control and decentralization, blockchain functionalities are preferred. It is noteworthy to emphasize that there is no one cryptographic primitive which is best suited to protect genomic data. However, finding a balance between the computation and communication efficiency of the proposed cryptographic technique to address the security problems in genomic data analyses and computational task is of utmost importance. In this thesis, we identify the following research problems:

1. To advance the field of genomic privacy, an investigation to clarify the state of knowledge and identify the research gaps in this domain is vital for the research community. Recent articles on genomic privacy, each with a different focus and contribution, have investigated the privacy attacks threatening genomic data and reviewed the techniques to mitigate these attacks [17, 7, 18, 19, 20]. However, these studies do not present a comprehensive examination and comparison of the current state-of-the-art in terms of their security goals, adversarial model and attacker background knowledge, and complexities (computation and transmission overheads). Therefore, there is the need for a comprehensive investigation and examination of state-of-the-art techniques to secure genomic data.

2. The traditional approach to health care delivery which is based on clinical symptoms and classic laboratory markers creates a situation where drugs and treatment work for some patients but leads to adverse drug effects in others [5]. This is due to the genetic heterogeneity amongst patients. Personalized medicine which tailors treatment based on a patient's unique genetic makeup aims to address this limitation [5]. However, genotype guided drug dosage prediction models expose patients sensitive genomic and clinical data, thereby creating

a security vulnerability which can be exploited by an attacker. Therefore the research question arises as to how patients genotypes and clinical data can be protected while allowing the execution of dosage prediction models?

3. Third-party personal genomics companies who sell DNA sequencing and genetic testing to their customers operate a business model where they collate DNA data from their customers and sell them to pharmaceutical companies for research into developing new drugs [21, 22]. However, this creates challenges in relation to access control, security and lack of compensation from third parties for monetizing their customers genomic data. A recent survey reveals that more than 50% of the US population would sell their genetic data for $95 with the right privacy measures in place [23]. Thus, there is a need to address genomic access control and privacy issues while allowing individuals to profit from sharing their genomic data.

4. The blockchain is an emerging technology with the functionalities of decentralization, transparency and immutability which is capable of revolutionizing genomic data management and access control [24, 25]. Though still in its early stages, this will lead to a paradigm shift in giving control to DNA sample donors to track their sequenced DNA and manage third parties who can view their data with fine-grained access control permissions. This is a fairly new research area and there are limited studies which investigate and examine the potential and integration of blockchain technologies in genomics and how the genomics landscape is changing. Hence, there is a need for a study to be conducted in this direction as it will serve as a guide for researchers and policymakers.

5. Federated learning (FL), is an emerging technology in artificial intelligence, provides model training confidentiality through distributed learning across multiple parties while their local training datasets are kept private [26]. A fundamental limitation of FL is that models trained on heterogeneous datasets have poor accuracy. This is a result of the model weight divergence problem between the uneven distributions of classes amongst the learning parties [27]. In addition,

FL is challenged by security issues as an adversary can carry out privacy attacks on shared gradients, and a lack of verifiability for aggregated gradients. These problems are exacerbated for FL for DNA sequence classification tasks since DNA sequences are highly heterogeneous and sensitive. What is worse is that the current state-of-the-art FL techniques to improve model accuracy on heterogeneous training datasets only achieve satisfactory improvements for image and natural language learning tasks but not for DNA sequence classification tasks [27, 28]. Hence, the research question is how to improve the accuracy of an FL model trained for DNA sequence classification task while protecting the privacy of shared gradients with the functionality of verifying aggregated gradients?

1.3 Research Significance and Contributions

This thesis contributes to the field of genomic privacy to solve the aforementioned research problems as follows:

1. First, this thesis contributes to the field of genomic privacy by investigating current privacy attacks on genomic data and examining the adversarial background information, inference and prediction techniques required to carry out these attacks. We discuss and evaluate cryptographic primitives utilized to design privacy-preserving genomic analyses techniques. We comprehensively investigate current state-of-the-art genomic privacy-preserving techniques and classify them based on their biomedical application domain into genomic aggregation, GWAS and statistical analysis, sequence comparison and genetic testing. In addition, we compare and evaluate these techniques based on their cryptographic primitives, security goals and computation and transmission overheads. Finally, the challenges and future directions in the field of genomic privacy are highlighted and discussed.

2. This thesis also makes an important contribution to personalized medicine to secure patients' sensitive genomic and clinical data used for genotype guided drug dosage prediction. We develop novel secure techniques using warfarin regression models, homomorphic encryption and secure two-party computation

(SMC) protocols to predict warfarin dosages specific to patients' genetic variants. To the best of our knowledge, our work is the first to develop a secure genotype-guided dosage prediction framework. The evaluation results on the datasets of patients with thromboembolic disorder demonstrate that our proposed approach is secure and efficient for real-world clinical settings, and the estimated warfarin dosages are similar to that of ground truth dosages with negligible losses.

3. Another significant contribution made by this thesis is the development of a novel blockchain genomic access control and variant discovery framework. Novel protocols are proposed using blockchain transactions and smart contracts to allow genomic data owners (DOs) to control and sell access to their data to genomic data users (DUs) for cryptocurrency. In addition, homomorphic computation with secure two-party protocol is proposed to enable DUs to securely run queries to discover DOs of interest. Query optimization techniques using genomic data partitions, binary search trees and bloom filters are also proposed for our blockchain variant discovery protocol. Our evaluation results demonstrate that our proposed framework outperforms current state-of-the-art.

4. This thesis contributes to the investigation and examination of the integration of blockchain functionalities for genomic and phenotypic data interoperability, access control and auditability. In addition a comprehensive review of blockchain architectures, storage techniques and consent models utilized by the current state-of-the-art genomic analyses techniques is presented. Furthermore, it investigates and categorizes state-of-the-art blockchain-based genomic and phenotypic data sharing and analytics techniques. A comparative analysis of these techniques is presented and evaluated in terms of their security requirements, blockchain functionalities and transaction complexities. Research gaps, challenges and future directions in this field are discussed which we believe will be useful for researchers and policymakers.

5. Lastly, this thesis contributes by proposing a new blockchain federated learning

(FL) framework to address the FL challenges of poor model accuracy, privacy attacks, and a lack of verification of aggregated gradients for DNA sequence classification. A new technique is proposed to improve model accuracy by minimizing the weight divergence between FL parties' training datasets. Also a new consensus algorithm (proof of aggregated gradients) is developed to allow parties to verify the model aggregation work of a block validator. The experiment results demonstrate that our approach yields superior model accuracy compared to current state-of-the-art techniques.

1.4 Thesis Outline

The rest of this thesis is organized as follows.

Chapter 2: This chapter presents a comprehensive literature review on the progression of genomic privacy and the cryptographic functionalities utilized to secure genomic data. Also the most relevant privacy attacks carried out on genomic data are investigated and their mitigation strategies are examined. In addition, current state-of-the-art genomic privacy-preserving techniques are investigated, classified and a comparative analysis is present. Finally, this chapter discusses the challenges and future directions in the field of genomic privacy.

Chapter 3: This chapter proposes a novel secure framework to predict drug dosage in personalized medicine based on a patient's genetic variants and clinical data. The novel secure techniques in this chapter are based on warfarin regression models, homomorphic encryption and secure two-party computation (SMC) protocols. The evaluation results demonstrate that our approach is secure and efficient in computation and communication.

Chapter 4: This chapter presents the development of a novel blockchain-based genomic access control and variant discovery framework. New protocols are proposed with blockchain transactions and smart contracts to enable genomic data owners to control access to their data. Furthermore, this chapter proposes a new variant discovery protocol to enable genomic data users to query genomic data owners with variants of interest. The evaluation results demonstrate that our proposed framework is effi-

cient and outperforms the current state-of-the-art.

Chapter 5: This chapter investigates how blockchain functionalities are integrated for genomic and phenotypic data interoperability, access control and auditability, and presents a comprehensive literature review. Furthermore, this chapter investigates and classifies state-of-the-art blockchain-based genomic and phenotypic data sharing and analytics techniques and presents a comparative analysis on their security requirements, blockchain functionalities and transaction complexities. Finally, the research gaps, challenges and future directions in this field are discussed.

Chapter 6: This chapter proposes a novel secure federated learning (FL) framework for DNA sequence classification. In addition, this chapter introduces new techniques to improve model accuracy, protect the confidentiality of model gradients and verify the integrity of aggregated gradients on the blockchain. The evaluation results demonstrate that our approach is efficient in computation and transmission throughput, and yields superior model accuracy compared to current state-of-the-art techniques.

Chapter 7: This chapter concludes our contributions and findings, and discusses future research directions.

Chapter 2: Ensuring Privacy and Security of Genomic Data and Functionalities

In recent times, the reduced cost of DNA sequencing has resulted in a plethora of genomic data which is being used to advance biomedical research, and improve clinical procedures and healthcare delivery. These advances are revolutionizing areas in genome-wide association studies (GWAS), diagnostic testing, personalized medicine and drug discovery. This, however, comes with security and privacy challenges as the human genome is sensitive in nature and uniquely identifies an individual. In this chapter, we discuss the genome privacy problem and review relevant privacy attacks, classified into identity tracing, attribute disclosure and completion attacks, which have been used to breach the privacy of an individual. We then classify state-of-the-art genomic privacy-preserving solutions based on their application and computational domains (genomic aggregation, GWAS and statistical analysis, sequence comparison and genetic testing) that have been proposed to mitigate these attacks and compare them in terms of their cryptographic primitives, security goals and complexities-computation and transmission overheads. Finally, we identify and discuss the open issues, research challenges and future directions in the field of genomic privacy. We believe this chapter will provide researchers with the current trends and insights on the importance and challenges of privacy and security issues in the area of genomics.

NOTE: The content of this chapter has been published in *Briefings in Bioinformatics.*

Mohammed Yakubu, A., & Chen, Y.-P. P. (2020). Ensuring privacy and security of genomic data and functionalities. *Briefings in bioinformatics*, 21(2), 511-526.

2.1 Introduction

The acquisition of genetic samples has become ubiquitous and is increasing in recent times due to the decrease in sequencing cost. These data are shared in public databases, biobanks and repositories (e.g., UK biobank [1] and the 1000 Genomes Project [3]) to assist researchers and clinicians to advance biomedical research to better understand the structures and functionalities of biological data - DNA, RNA and proteins. This advancement in research will, amongst many other things, broaden the use and applications of genetic testing in areas such as (i) paternity testing- to determine whether two individuals are parent and child, (ii) diagnostic testing- to diagnose whether an individual is affected by a certain disease, and (iii) pharmacogenetic testing- in personalized medicine to tailor treatment for an individual, and the development of efficient drugs and therapies.

Despite these benefits, there are still serious concerns about the security and privacy of genomic data in storage, sharing, in transit and during computation. Donors of DNA samples sometimes ask questions such as: "How is my genetic data stored? Who has access to it? What security measures are in place to protect my privacy?" These concerns are born out of the fact that:

1. In Australia, genetic data is collected and used to drive the agenda of the Australian Genomics Health Alliance (AGHA) in genomic medicine [29].

2. A new bill moving through the US congress, if passed, will allow companies to have access to their employee's genetic data by requiring them to undergo genetic testing or they will be fined thousands of dollars [30].

3. In the UK, the National Health Service (NHS) has proposed a genetic database where the DNA of babies will be sequenced at birth. [31].

The human genome is special and has certain characteristics such as being unique, it does not change much over an individual's lifetime, it is non-revocable, it reflects ethnic heritage, is correlated between relatives and can identify predisposition to diseases which makes it a vast collection of sensitive information and prone to privacy

risk and attacks [7]. As a result of these sensitive properties, there are ethical regulations on how genomic data should be shared. Federal laws in countries such Australia, Canada, USA and UK prohibit health insurance companies and employers from discriminating against people based on genetic information. However, there are still loopholes in these laws, for instance, in some states in the US, this law does not apply to companies providing disability, life, or long-term care insurance [32]. Furthermore, companies still discriminate without openly stating their reasons for doing so [33]. This is however different in the EU with the new General Data Protection Regulation (GDPR) law where the use of personal data has been restricted to the purpose for which it was collected [34]. For instance, this prohibits processing of direct-to-consumer (DTC) data for other purposes such as insurance without consent from individuals. Examples of such discriminations are (i) an employer denying an employee a promotion because his DNA reveals that his skillset does not match the job profile, and (ii) an insurance company declining coverage for an individual based on his inclination to certain health conditions (e.g., breast cancer, sickle cell anemia and other Mendelian diseases) revealed from his DNA. Another privacy risk is compromising kinship privacy. This occurs when leaking an individual's genome gives away some genetic information about his close relatives due to DNA correlation amongst individuals from the same family [35, 12, 36]. One of the paramount implications this has is that people are skeptical about privacy issues related to genomic data and reluctant to participate in genetic testing programs which is crucial in helping researchers learn the structure and functions of the human genome for the advancement of areas in biomedical science such as personalized medicine. This is a global problem, not just a one-country issue. Thus, there is the need for the privacy, security and confidentiality of genomic data to be protected at storage and during computation. Genomic privacy is a multidisciplinary area and to address these issues, policymakers, and cryptographic and bioinformatics research communities are collaborating to enact policies and design secure frameworks which are yet to sufficiently address security goals and computational issues specific to genomic practical use-cases.

Related surveys and articles on genomic privacy, each with a different focus and

contribution, have discussed the privacy and security issues facing genomic data and surveyed proposed mitigation techniques (cryptographic and non-cryptographic). Techniques applicable for compromising and breaching the privacy of genomic data and mitigation methods have been reviewed in [17]. Another article also discussed the genome privacy problem, privacy attacks and mitigation strategies [7]. This study further presented (i) an opinion poll from the biomedical community, and (ii) a framework in the context of health care, biomedical research, legal and forensics, and DTC services, for the security and privacy of genomic data. Privacy and security problems for sharing, storing and computing on genomic data with related query and output privacy guarantees are presented in [18], and techniques applicable to addressing these problems are reviewed. A categorization of genome privacy problems and their solutions based on sequence alignment and querying private and public genomic databases is presented in [37]. Other surveys have focused on discussing genomic privacy within a specific jurisdiction. For instance, a study discussed the technical and ethical aspects of genomic privacy focused on data sharing for biomedical research in the United States [19]. A recent study proposed a methodology for the categorization of privacy techniques which is then used as a basis for the evaluation and critical analysis of challenges confronting the genomic privacy community [20]. In contrast to these useful surveys, our work focuses on (i) inspired by the categorization of privacy breaching techniques presented in [17], we discuss the adversarial goal, background knowledge and inference technique required for each class of attack as presented in Table 2.1, and (ii) we evaluate and compare privacy-preserving techniques applicable to mitigating these attacks in terms of their security goals (confidentiality, integrity, query privacy, output privacy, and adversarial model) and complexities (computation and transmission overheads) which is not addressed in previous surveys. The evaluation criteria (security goals and complexities) we use in this chapter are essential in determining the scalability and feasibility of a genomic privacy-preserving technique in a real-life setting. In summary the goals of this chapter are as follows:

1. We present an overview of current privacy attacks on genomic data and discuss the adversarial background information, inference techniques and predictions

required to compromise the privacy of an individual in each attack category.

2. We discuss and compare cryptographic primitives and how their native overheads, strengths and weakness can influence the design of genomic privacy techniques.

3. We classify genomic privacy-preserving techniques based on their biomedical, clinical and research use-cases into genomic aggregation, GWAS and statistical analysis, sequence comparison and genetic testing, and present a comparative analysis using their cryptographic primitives, security goals and complexities (computation and transmission overheads).

4. We also discuss the challenges, research gaps and future directions in the field of genomic privacy and highlight why they should be addressed.

The remainder of this chapter is organized as follows: Section 2.2 presents the most relevant current privacy attacks on genomic data which are categorized into identity tracing, attribute disclosure and completion attacks. Sections 2.3 and 2.4 discuss the cryptographic building blocks, and security goals and systems architecture employed in protecting genomic data respectively. Section 2.5 surveys state-of-the-art genomic privacy-preserving solutions and compares them in terms of application domain and complexities – computation and transmission. Open issues and challenges in genomic privacy research domain are presented in Section 2.6 and the work is concluded in Section 2.7.

Table 2.1: State-of-the-art privacy attacks on genomic data

Work	Adversary goal	Adversary background knowledge	IT	IC
Identity tracing attacks				
Sweeney et al., 2013 [38]	Re-identify participants of the personal genome project	Victim's demographic data	Demographic data matching	M
Gymrek et al., 2013 [39]	Triangulate identity from surnames and Y chromosomes	Victim's surname, Y-chromosome haplotypes and demographic data	Statistical hypothesis testing	M
Shringarpure et al., 2015 [40]	Re-identifying individuals and their relatives within a beacon	VCF file of the victim's genome, # of individuals in the beacon, SFS of the population in the beacon	Likelihood-ratio test	L
Raisaro et al., 2017 [41]	Re-identifying individuals and their relatives within a beacon	VCF file of the victim's genome, # of individuals in the beacon, AFs in the beacon	Likelihood-ratio test	L
von Thenen et al., 2018 [42]	Re-identify an individual within a dataset in a beacon	VCF files of people from the victim's population, corresponding MAF and LD, and high-order correlation	High-order Markov chain model	H
Erlich et al., 2018 [13]	Re-identify an individual by long-range familial search	Victim's genotype, familial relations, demographic data (location, age, and sex)	DNA matching, search space pruning	L
Attribute disclosure attacks				
Homer et al., 2008 [43]	Determine the presence of an individual in a GWAS	Victim's SNP profile, GWAS statistics: large set of SNPs ($> 10,000$)	Statistical testing, distance measure	M
Fredrikson et al., 2014 [44]	Infer sensitive genetic markers of an individual from a warfarin dosage pharmacogenetic model	Warfarin dose predicting model, victim's demographic data and dosage	Model inversion	M
Humbert et al., 2015 [45]	Predict an individual's predisposition to Alzheimer's disease	Victim's anonymized genotype and phenotype, SNP-trait association	Statistical methods based on SNP correlation	M
Cai et al., 2015 [46]	Determine the presence of an individual in a GWAS	Victim's SNP profile, GWAS statistics: small set of SNPs (> 25)	Data mining and matching techniques, deterministic proofs of study inclusion	L
Lippert et al., 2017 [47]	Predict an individual's traits for re-identification	Victim's genotype and phenotypes	Machine learning methods	H
Completion attacks				
Kong et al., 2008 [48]	Infer haplotypes of ungenotyped individuals from their relatives genetic information	Family members pedigree structure, genotypes (LD between haplotypes)	Genotype imputation, haplotype sharing graph	H
Humbert et al., 2013 [35]	Infer an individual's genotype from their relatives genomes	Family members familial relationships, genotypes (LD between SNPs, Minor allele frequencies)	Belief propagation, factor graph	L
Humbert et al., 2017 [12]	Infer an individual's genotype from their relatives genomes and phenotypes	Family members familial relationships, genotypes (LD between SNPs, Minor allele frequencies) and phenotypes	Belief propagation, factor graph	L

Continued on next page

Table 2.1 State-of-the-art privacy attacks on genomic data - Continued

Deznabi et al., 2017 [49]	Reconstruct missing parts of an individual's genotype	Pedigree structure, victim's partial genotype (high order correlation between SNPs) and phenotypes	Belief propagation, factor graph	M
He et al., 2018 [14]	Predict the genotypes and traits of individuals based on publicly available genome data and traits released by individuals or their relatives	Genotypes and phenotypes of the victim and relatives, and SNP-trait association from GWAS	Belief propagation, factor graph	L

Notations: **H**: High; **IT**: Inference technique; **IC**: Inference complexity; **L**: Low; **LRT**: Likelihood-ratio test; **M**: Medium; **SFS**: Site frequency spectrum

We measure the inference complexity (level of hardness to perform) of a technique as a function of the genetic and prior knowledge of the adversary, amount of data processing and molecular techniques involved [17]. Inference technique is of low complexity if adversary has genetic knowledge, adequate prior knowledge with low data processing. Medium complexity techniques require genetic knowledge, adequate prior knowledge and medium data processing. High complexity techniques also require genetic knowledge, adequate/little prior knowledge and large-scale data processing

2.2 Privacy Attacks on Genomic Data

The privacy risk presented in the introduction of this chapter together with some real-life privacy compromising scenarios on genomic data discussed in [7] have motivated the cryptographic and bioinformatic research community to dedicate an area of research with the aim of better understanding the privacy risks related to genomic data. In this area, researchers quantify privacy loss and explore attacks on genomic data by modeling attacker behaviors as they would have occurred in real-life. A privacy attack occurs when an adversary compromises the privacy of an individual by exploiting sensitive information inferred from his/her DNA for purposes such as personal gain, blackmail, to alter evidence in the case of forensics to be used in the court of law or some other dubious reasons. The adversary in this context uses public information (e.g., genomic and phenotypic data, demographic data, genealogy and family pedigree) gathered from genome-sharing websites (e.g., PatientsLikeMe and OpenSNP) and online social networks to (i) triangulate the identity of an individual, (ii) infer sensitive attributes such as disease and drug abuse, or (iii) reconstruct an

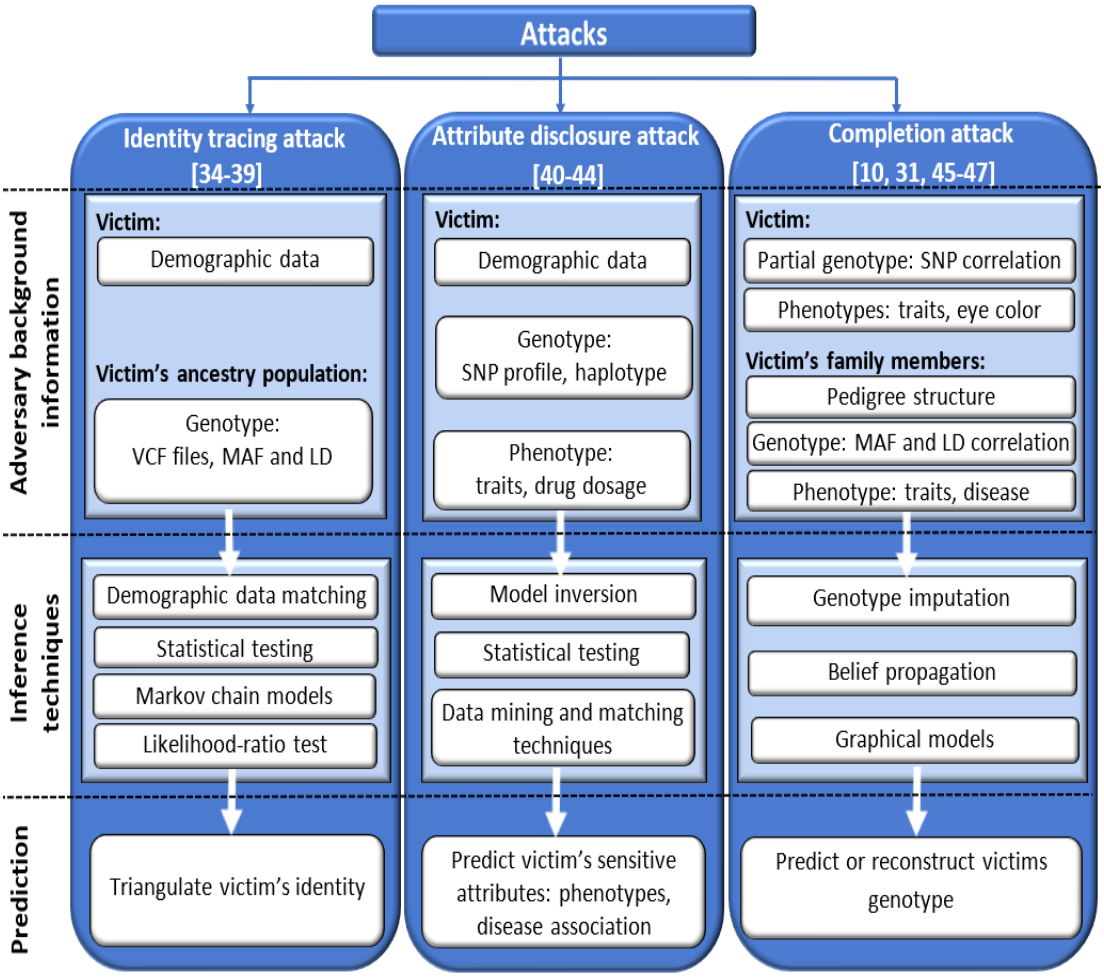


Figure 2.1: Taxonomy of genomic privacy attacks. A typical genomic privacy attack (identity tracings, attribute disclosure and completion attacks) involves three steps. First, the adversary acquires background information about the victim and his/her family members if required. The adversary then applies inference techniques and finally predicts the identity or disease associations of the victim.

individual's genetic sequence from partially available DNA data. In the rest of this section, we briefly discuss privacy attacks on genomic data which we categorize into the three most relevant attack types - identity tracing attacks, attribute disclosure attacks and completion attacks. We present the current state-of-the-art attacks on the privacy and security of genomic data in Table 2.1, and represent a taxonomy of the attacks in Figure 2.1 which depicts the background information and techniques required to launch each attack. In this and coming sections we use "victim" to refer to the individual whose genome is compromised and "adversary" to indicate the individual carrying out the attack.

2.2.1 Identity Tracing Attacks

An identity tracing attack occurs when an adversary obtains an anonymized or de-identified DNA and identifies the owner by using publicly available quasi-identifiers (age, sex, surname, zip code, etc.) obtained from social media or public record search portals such as PeopleSmart and FindOutTheTruth [17]. This was demonstrated in a study where participants of the Personal Genome Project (PGP) were re-identified using demographic data - birth dates, sex, zip code [38]. Another study discovered that an adversary can infer surnames from Y chromosomes acquired from genealogical websites e.g. Ysearch (www.ysearch.org) and SMGF (www.smgf.org) [39]. These surnames were then combined with other types of metadata, such as age and state, to triangulate the identity of the victim.

Amongst their goals to drive the agenda for secure genomic data sharing, the global alliance for genomics and health (GA4GH), which creates policies and standards for the secure sharing of genomic data, initiated a project called beacon. A beacon is a web service that allows institutions to implement and securely share genetic data, and answer queries such as "Do you have an allele at a specific position in a genome?" and respond with a "Yes" or "No" [50]. The security goal of the beacon is to make it difficult for an adversary to re-identify an individual within a dataset. However, recent studies have demonstrated that beacons are more vulnerable than expected and have highlighted the severity of the genomic privacy risk [40, 42]. These studies launched an attack to re-identify an individual by querying the beacon and inferring the results and alleles at certain positions. A recent study [41] further examined the risk associated with beacons and the attack proposed in [40]. This study proposed an attack that considers an adversary with some background knowledge about the allele frequencies (AFs) in the targeted beacon and as a result yields a higher re-identification power than [40]. In addition to re-identification, datasets with disease(s) associations makes it possible to infer the victim's disease status. In forensics, crime solving has been enhanced by identifying and tracing suspects via distant familial relatives. The suspect's distant genetic relatives are acquired by looking up his genome-wide profile against a public third-party consumer genealogy service provider such as GEDmatch.

A recent study demonstrated that such techniques could be exploited by an adversary to comprise the privacy of an individual by using demographic information such as location, age, and sex [13].

2.2.2 Attribute Disclosure Attacks

The goal of the adversary in attribute disclosure attacks is to predict sensitive attributes of the victim, such as phenotypes, disease association and drug abuse. In this attack, the DNA sample of the victim is known and the adversary matches it against public genetic study databases or published GWAS results and statistics [17].

Summary statistics have long been thought to conceal the identity and associated phenotypes of GWAS participants. However, research within the past decade has proved this notion to be incorrect. One of the first work to demonstrate vulnerabilities in aggregate statistics showed that an adversary with access to the victim's SNP profile can infer the presence of the victim in a case group of a GWAS study from allele frequencies of a large set of SNPs. A recent study in this line of research also exploited summary statistics by using allele frequencies to determine an individual's presence in a GWAS and hence their disease status [46]. This study also showed that an adversary can yield higher prediction confidence with a smaller set of SNPs (> 25 SNPs) as opposed to past studies [43, 51] which required large amounts of SNP data ($> 10,000$ SNPs and > 200 SNPs) to make a prediction. In addition to exploiting GWAS statistics for disease prediction, visible phenotypic traits such as eye color can be linked to public genomic databases to re-identify an individual's genotype and subsequently infer disease associations. This was proven in a study where researchers used visible phenotypic traits to infer an individual's predisposition to Alzheimer's disease [45]. In another study, authors proposed a technique for predicting multiple phenotypic traits (face, voice, age, height, weight, BMI, eye color, and skin color) from whole genome sequenced (WGS) data [47]. These predictions (combination of multiple DNA-based predictive models) were later used to re-identify individuals from a subset of their study cohort with good accuracies reported. However, these findings received criticisms from fellow researchers in the field. An example of such critiques

argued that face structure inference is driven by population averages and not from trait-specific markers [52]. They further demonstrated a base-line re-identification procedure with similar accuracy as [47] that relies on low dimensional demographic information: age, sex, and self-reported ethnicity which are expected to be available to the adversary as opposed to some phenotypic data (high dimensional trait) used in [47] which might not be available to the adversary.

In a personalized medicine application where machine learning models (pharmacogenetic models) are trained to predict warfarin dosage, researchers have proven that sensitive information leaked by the models can be used to launch an attack to infer the sensitive genetic markers of an individual as well as some demographic data - age, race, height, and weight [44].

2.2.3 Completion Attacks

Completion attack is the application of techniques such as genotype imputation to reconstruct the victim's genetic information from partially available DNA data or a family member's DNA sequence [17, 53]. Studies have demonstrated that this attack is commonly used to breach kin privacy where the privacy of the victim is compromised by exploiting the genetic information of his family members obtained from public online resources: online social networks (OSNs), genome-sharing websites (e.g., OpenSNP, 23andMe and PatientsLikeMe) and genealogical data repositories [35]. This is possible because of the high correlation of genomic data between family members which leads to interdependent privacy risks. This has been demonstrated in a study where by observing the genome of an individual it is possible to reconstruct the genomes of relatives using statistical relationships (pairwise correlation) between DNA sequences [35]. An extension to this work improved the inference power by adding additional background information of the relatives phenotypes [12]. Another study [49] to improve the inference attack in [35] considered complex correlation in the genome as opposed to pairwise correlation used in [35]. In addition to the aforementioned reconstruction attacks, a recent study also predicted the victim's genotypes and phenotypes [14]. This technique, however, yielded low computation complexity

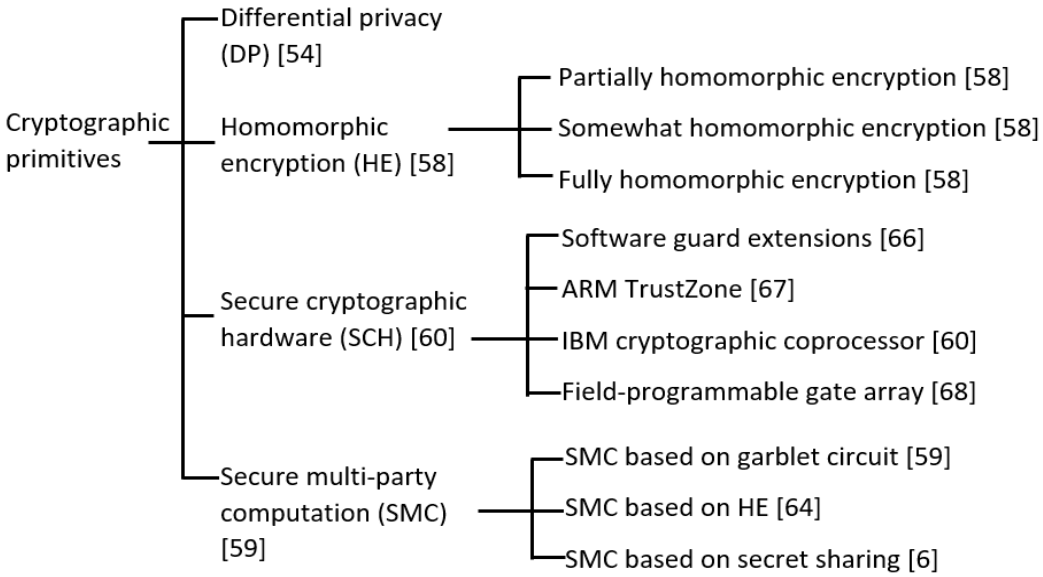


Figure 2.2: Categorization of cryptographic primitives applied in literature to secure genomic data.

as compared to previous studies.

The above discussed attacks have relied on SNP correlations for inference, however it is possible to reconstruct a victim's genome by observing identical regions of the DNA sequence between his relatives. In a study to demonstrate this, haplotypes for individuals who are not genotyped are identified by observing DNA regions that are identical by descent (IBD) between their close and distant relatives [48].

2.3 Cryptographic Primitives

Cryptographic primitives are low-level cryptographic algorithms used to build cryptographic systems to provide information security. There are several primitives in the cryptographic literature; however, in this section we discuss only those that are relevant and have been used to secure genomic data for storage and computation. Categorization of these techniques are illustrated in Figure 2.2. In this and coming sections we use "efficiency" to refer to computation and communication complexities.

2.3.1 Differential Privacy (DP)

It has been proven that an individual's privacy can be compromised from releasing aggregate statistics such as minor allele frequencies (MAFs), chi-square (χ^2) and p-

values used in genome-wide association studies (GWAS) [43, 54, 46]. This discovery led the National Institutes of Health (NIH) to remove statistics from public genomic databases and revise its policies on how genomic data should be shared [55]. To this end, differential privacy was proposed, which is a probabilistic notion of privacy, to provide a statistical measure of privacy. It guarantees that the addition or removal of a record from a database does not substantially affect the outcome of any analysis [56]. Let D_1 and D_2 be two databases which differ in at most one row and S be a possible set of query outputs on the databases, then a randomized algorithm A is ε-differentially private over D_1 and D_2 if the following holds:

$$Pr[A(D_1) \in S] \leq e^{\varepsilon} \times Pr[A(D_2) \in S] \tag{2.1}$$

where ε is the privacy parameter which in most cases is achieved by adding random noise to the output S. The addition of noise reduces the accuracy of the computation. As a result, there is often a trade-off between the privacy and utility (useful outputs) of the data. In applications where GWAS is used to understand the correlation between genetic regions (loci) and drug response/reactions for the purpose of advancing pharmacogenomics, accuracy is critical and should not be compromised [57]. Recent studies have discussed ways to approach this trade-off to yield near accurate results [58, 59]. However, the issue of accuracy is still an active research area and there is still a need to find the right balance between computational efficiency, privacy loss and the statistical utility of perturbed p-values, χ^2-statistics, MAFs etc.

2.3.2 Homomorphic Encryption (HE)

An encryption scheme is homomorphic if it preserves certain structures which allows arithmetic operations to be performed directly on its ciphertext. It can be represented by Equation 2.2, where $\circ_p$ and $\circ_e$ are operations on plaintext and ciphertext respectively.

$$\forall m_1, m_2 \in M, \quad D(E(m_1) \circ_e E(m_2)) = m_1 \circ_p m_2 \tag{2.2}$$

An encryption scheme can be homomorphic on addition or multiplication depend-

ing on whether $\circ_p$ is the addition or multiplication operator. Based on the operations that can be computed of their encrypted data, homomorphic encryption techniques can be categorized into (i) partially homomorphic encryption: allows either addition or multiplication, (ii) somewhat homomorphic encryption: allows both addition and multiplication but limited in number of computation times, and (iii) fully homomorphic encryption: supports both addition and multiplication for unlimited number of computations. A detailed discussion on these categories is beyond the scope of this chapter and can be found at [60]. In preserving the privacy of genomic data, homomorphic encryption is useful when data owners want to outsource their data to a public cloud while allowing certain computations such as paternity tests and disease risk tests to be carried out on encrypted single nucleotide polymorphisms (SNPs), haplotypes or short tandem repeats (STRs).

Table 2.2: Comparison of cryptographic primitives

Techn.	**Operations**	**HR**	**Overheads**				**Advantages**	**Limitations**
			CP	**CM**	**ST**	**AL**		
DP [56]	Any operations	Any CPU	M	L	M	H	Provides privacy of individual data within a statistical database, supports any mathematical operation, provides privacy against a computationally unbounded adversary	High accuracy losses, degraded data utility
HE [60]	Addition and multiplication	Any CPU	H	M	H	M	Allows computation over encrypted data, provides at most semantic security, ensures input and output privacy of data	Computationally expensive, restricted to addition and multiplication operations, cipher-blow up problems leading to higher storage cost
SMC [61]	Boolean operations	Any CPU	M	H	M	M	Allows private computation over multiple parties, provides unconditionally or information-theoretic security, ensures input privacy of data	High data transmission cost, scalability issues
SCH [62]	Any operations	CC	L	L	L	L	Efficient computation and communication with negligible accuracy losses, supports any mathematical operation, provides security against malicious adversary, ensures integrity of data	Restricted in memory size which depends on hardware and OS

Notations: **AL**: Accuracy loss; **CP**: Computation; **CM**: Communication; **CC**: Cryptographic coprocessor; **DP**: Differential privacy; **HE**: Homomorphic encryption; **HR**: Hardware requirement; **H**: High; **L**: Low; **M**: Medium; **ST**: Storage; **SMC**: Secure multi-party computation; **SCH**: Secure cryptographic hardware; **Techn.**: Technique;

2.3.3 Secure Multi-party Computation (SMC)

SMC is used in a setting where multiple parties $P_1, P_2, ..., P_n$ need to collaborate to compute a function $f(.)$ on their data $D_1, D_2, ..., D_n$ while keeping them private except for the computation result $f(D_1, D_2, ..., D_n)$.

The most popular way of implementing SMC is using garbled circuit (GC) which was first introduced by Yao [63]. The GC implementation expresses the function $f(.)$ as a Boolean circuit (a collection of gates) where computation is performed for each gate which involves multiple rounds of communication between parties. Generally speaking, this protocol is expensive in communication for genomic data which often has millions of nucleotides - i.e. the number of iterations is linear to the sequence length. SMC-based privacy-preserving techniques for processing genomic data usually employ a two-party computation case to tackle simple tasks such as sequence comparisons, genetic testing and GWAS, yet they have not been able to get around the complexity issues [64, 65, 59]. Other efficient implementations of SMC have used homomorphic encryptions and secret sharing in place of the expensive GC computations [66, 6].

2.3.4 Secure Cryptographic Hardware (SCH)

Secure cryptographic hardware uses hardware to complement software for data encryption and protection. It is most often implemented as part of the processors instruction set and comes in the form of cryptographic coprocessors, accelerators, chip cards, smart cards etc. The following are well known implementation of SCH:

- Software Guard Extensions (SGX) [67, 68]: SGX was introduced in 2015 by Intel as an extension to its 6th generation of core microprocessors. It allows an application to execute code and protect data within its own trusted execution environment (TEE) - a secure container called an enclave. Contents of the enclave are sealed and protected against external software such as malicious software (privileged malware) and privileged software (virtual machine monitors, BIOS, or operating systems).

- ARM TrustZone [69]: TrustZone is a security extension provided by ARM in some of their processors (ARM1176, Cortex-A series, v8-M architecture) which creates two environments (a secure world and non-secure world) that can run simultaneously on a single core. The secure world provides confidentiality and integrity to high-value code and data such as cryptographic operations.

- IBM cryptographic coprocessor [62]: this is a hardware security module consisting of a microprocessor, memory, special cryptographic hardware and a random number generator contained in a tamper resistant box. It provides a secure environment where data and cryptographic functions are processed and sealed from the rest of the server.

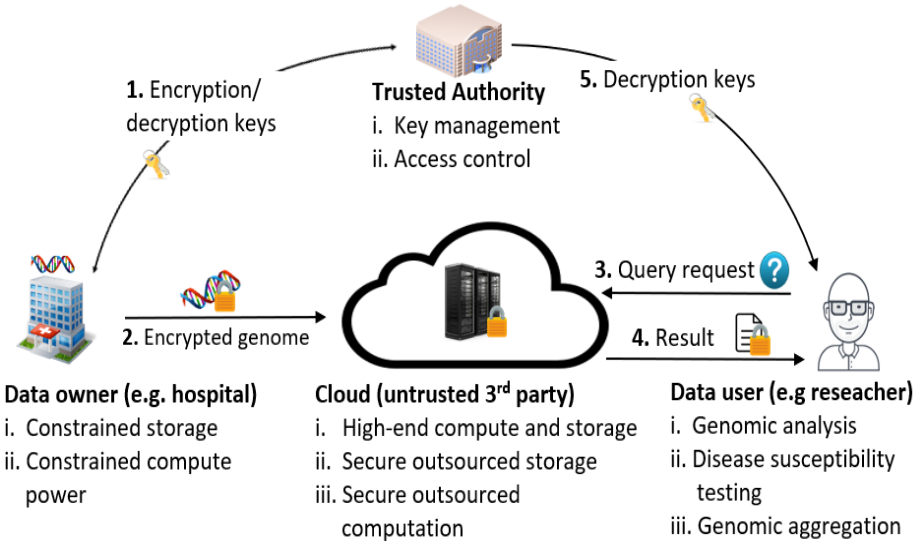
- Field-programmable gate array (FPGA) [70]: FPGA is a re-programmable integrated circuit that allows customers to tailor it to their needs. Similar to the SGX and IBM cryptographic coprocessor, FPGAs also have the capability of isolating the computation from the rest of the server.

Using SCH comes with benefits such as (i) it is more efficient than traditional cryptographic methods, (ii) it can compute any arbitrary function as opposed to HE techniques which are limited to addition and multiplication, and (iii) it is tamper resistant. Due to these advantages, they have been used to implement efficient and tamper-resistant secure genomic techniques [71, 72, 73, 74]. They are, however, restricted in memory size especially for processing genomic data with millions of nucleotides.

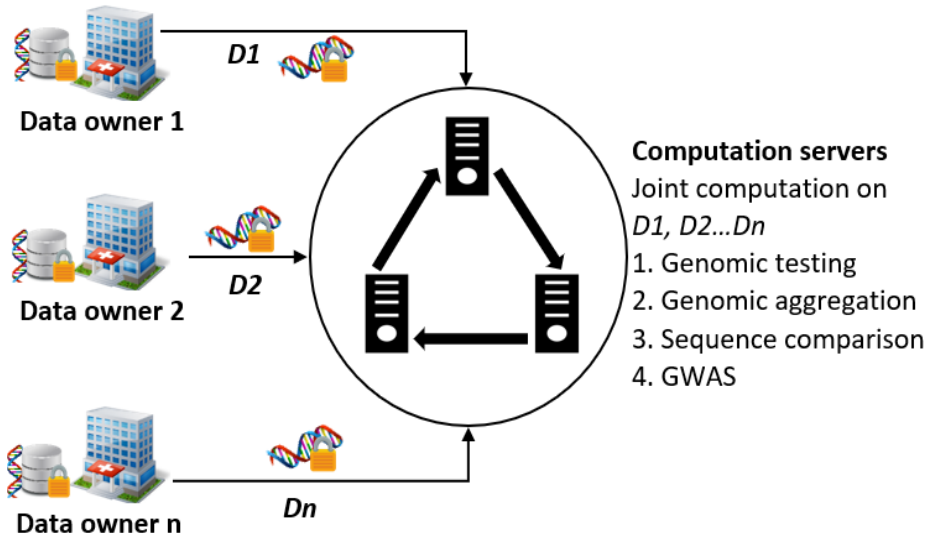
2.3.5 Choosing the Right Cryptographic Primitives

The selection of an encryption technique to protect the privacy of genomic data is vital in determining the scalability and feasibility of an application. The criteria for selection depends on a number of factors such as overheads (computation and communication) incurred by the cryptosystem, arithmetic operations and computational task supported by the encryption technique (e.g., linear operations are supported by HE, while comparison operations can be accomplished with SMC), the genomic ap-

plication use-case, the security requirements and threat model for the given scenario. It is noteworthy to mention that one or more encryption techniques can be combined to complement each other in a given framework. For instance, (i) in a genetic testing application where disease markers are tested and compared against encrypted genomes, HE is used for linear computation on encrypted data while SMC is utilized for comparisons [66], and (ii) DP has been combined with SMC to hide GWAS participants for the secure computation of meta-analysis [59]. We present a comparison of the various cryptographic primitives discussed in this Section in Table 2.2. We compare their overhead in terms of (i) computation time: the time taken to encrypt and decrypt a genome sequence [75], (ii) communication overhead: the amount of data bits transfered between parties in the cryptographic protocol [75], (iii) storage overhead: storage overhead or blowup is estimated as the difference in size between the encrypted data and its original plaintext version [75]. In designing techniques to protect genomic data, storage overhead needs to be minimized as the size of a human sequenced genome can be approximately 200GB depending on the coverage, number of reads and read length. Most encryption schemes, after converting data from plaintext to ciphertext increase the size of the ciphertext and as such, the resulting encrypted genome requires more space ($>$ 200GB) than its plaintext version. This is as a result of using larger encryption keys within a Galois field (finite field), and (iv) accuracy loss: the percentage of error margin between the results of plaintext genomic computations and its secure version [75]. i.e. $\frac{e}{G(D)} \times 100$, where e is the error margin and $G(D)$ is the plaintext genomic computation function on sequence data D. In DP applications, the amount of accuracy loss is dependent on the privacy parameter ε in Equation 2.3.1. Smaller ε means higher accuracy loss (higher noise variance) and stronger privacy, and vice versa. This often depends on trade-off between data privacy and utility. There are techniques on minimizing accuracy loss in DP. This however is a broad area and outside the scope of this chapter. We refer the reader to [56, 76]. Table 2.2 also presents the operations they support, the hardware requirement for their implementation and some limitations. We believe this will assist researchers in choosing the right cryptographic primitive/s when securing genomic data.



(a) Secure outsourcing



(b) Secure collaboration

Figure 2.3: System architectures adopted for protecting genomic data. (a) Secure outsourcing: data owner delegates storage and computation of genomic data to an untrusted 3rd party. ***Step 1***- data owner (DO) acquires encryption keys from a trusted authority (TA). The role of TA could be played by NIH. ***Step 2***- DO encrypts genomic data with the keys and transits it to the cloud for storage and computation. ***Step 3, 4***- at any point in time, a data user (DU) such as a researcher or pharmaceutical company can submit a query request to the cloud, for example, to query some biomarkers for a specific disease. The cloud then processes the query on the encrypted genomic data and sends back an encrypted result. ***Step 5***- finally, DU decrypts the results with the keys acquired from the TA. (b) Secure collaboration: this figure shows the application of SMC to facilitate the joint computation of genomic data. Multiple DOs encrypt their data and send them to centralized or distributed computation servers where genomic data processing such as GWAS is performed. DOs learn nothing about each other's data except for the computation results.

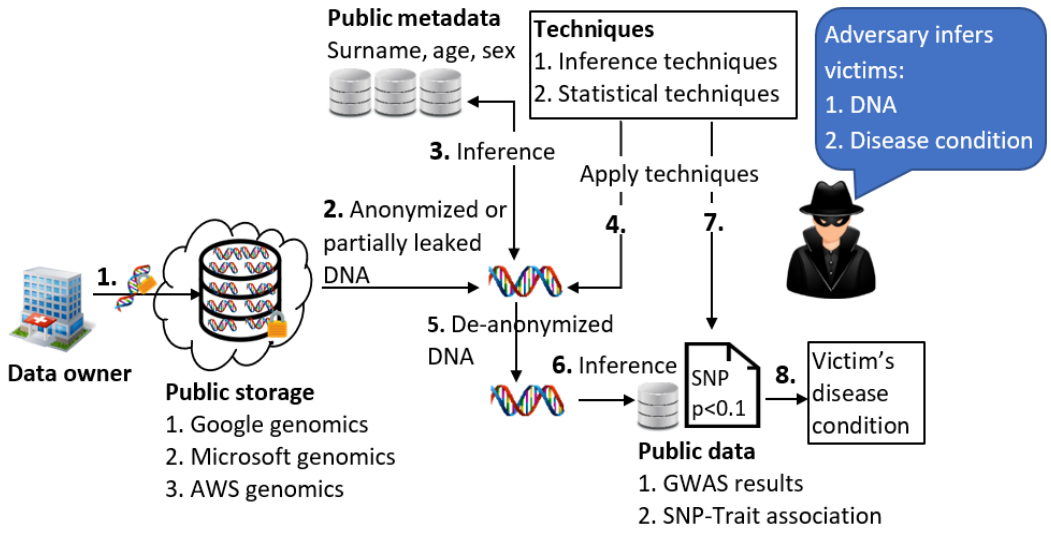


Figure 2.4: Privacy attack scenario: this is a typical attack case which can be carried out on either Fig. 2.3a or Fig. 2.3b depending on the amount of information leakage or compromised security parameters. ***Step 1***- DO encrypts or anonymizes data and sends it to a public server for storage. ***Step 2, 3, 4, 5***- the adversary acquires anonymized DNA or leaked information from the public storage and de-anonymizes it with a combination of metadata inferences plus some other inferences and statistical techniques. ***Step 6, 7, 8***- with the re-identified DNA, the adversary is capable of performing further statistical techniques coupled with lookups against a public dataset of GWAS results or SNP-trait association to predict the disease status of the compromised victim.

2.4 System Model And Architecture

In this section, we discuss the architectural framework types and models applicable to protecting genomic data.

2.4.1 System Architecture

In the genomic privacy literature, secure genomic techniques are built based on either of the following architectural or system models [77, 78].

***Secure outsourcing*:** In this architecture, as shown in Figure 2.3a, the data owner (e.g., a hospital or a research facility) is limited in resources and wants to securely outsource the storage and computation of genomic data to a third-party cloud (e.g., Google Genomics) while making data available to other stakeholders such as pharmaceutical companies and research labs. Homomorphic encryption is mostly used in this architectural framework where computation is performed on encrypted data in the cloud.

***Secure collaboration*:** Here multiple genomic data owners (e.g., hospitals, health institutions, research institutions and laboratories) want to jointly perform a function on their private data $D_1, D_2, ..., D_n$ while keeping them hidden from each other as depicted in Figure 2.3b. For instance, a research institution might want to perform a GWAS association test across multiple genomic datasets under different security jurisdictions. The main cryptographic primitive employed in this architectural setting is secure multi-party computation (SMC).

While we have stated the most commonly used cryptographic technique used under each model, it is important to note that depending on the use-case and application goals any cryptographic technique can be used under either models. For instance, (i) for a use-case to secure GWAS aggregate statistics DP has been employed under both models [59], and (ii) for a malicious setting SCH has been used [73]. It is also worth mentioning that both models can be combined into one framework to complement each other and not necessarily as standalones [79].

2.4.2 Security Model and Requirements

A security model and requirements is a set of assumptions and standards used as guidelines by researchers to design secure frameworks. In this subsection, we discuss the commonly used security guidelines adopted by the genomic privacy research community for modeling secure frameworks.

Threat Model

Threat models are used to describe the behavior of adversaries in a privacy-preserving framework. In genomic privacy and other privacy-preserving techniques, adversarial behaviors are categorized into semi-honest and malicious behaviors.

***Semi-honest model*:** This model is the most commonly adopted threat behavior for designing secure genomic techniques. It describes a security setting where parties faithfully follow the protocol specifications, meaning that they correctly carry out the computation and return correct results to the client [80]. They, however, try to learn sensitive information by observing computation and output results. For

example, direct-to-consumer genetic testing service providers such as 23andMe and color genomics will carry out the test (e.g., genealogy, ancestry or disease risk test) faithfully, but might try to learn additional information about the test participant by observing their SNP or STR profiles.

***Malicious model*:** This model addresses a more realistic security setting than the semi-honest behavior as parties can deviate from the protocol specifications to learn sensitive information [80]. An adversary in this threat environment can tamper with the genomic computation by injecting false data or returning false results to the client. Protocols that address this model provides more security but are generally more expensive than those build for semi-honest only [66]. This is as result of using other techniques such as zero-knowledge proofs to ensure that parties in the protocol don't behave maliciously.

Security Guarantees

In addition to systems architecture and threat models, a secure framework should achieve certain security goals and guarantees. This is mainly more application specific and for secure genomic techniques, the following guarantees ought to be achieved.

- Confidentiality: The genomic data should not be revealed to unauthorized parties.
- Integrity: The data owner or client should be able to verify any tampering done to the genomic data by an adversary.
- Query privacy: This ensures that query content sent to genomic datasets or biobanks is not learnt by unauthorized parties. This is useful in genetic testing and personalized medicine setting where pharmaceutical companies want to query a public genomic dataset while at the same time keeping the content of their proprietary test query private.
- Output privacy: The output of a genomic computation should not be revealed to any party other the intended recipient or the client.

2.5 Secure Techniques for Genomic Data

In this section, we survey the techniques that have been proposed to (i) mitigate the attacks discussed in Section 2.2, and (ii) protect the privacy and security of the human genome during storage and computation. For better organization and clarity, we group these techniques by their application domain - genomic aggregation, GWAS and statistical analysis, sequence comparison and genomic testing. Finally, we compare and evaluate them based on their security goals and complexities in terms of computation and communication.

2.5.1 Secure Genomic Aggregation

Aggregate operations (e.g., SNP or allele counts, frequencies, etc.) on large genomic datasets are the fundamental building blocks for genomic analysis such as GWAS, disease susceptibility tests and paternity test. For instance, a count query is used in disease susceptibility to test for rare variants (biomarkers) within a patient's DNA sequence and GWAS applies statistical analysis to allele frequencies to study gene-disease associations. These operations present an opportunity for genomic datasets from multiple sources and different jurisdictions, which otherwise cannot be shared on a common repository due to privacy issues, to be represented in the form of summary statistics often utilized by GWAS.

Several techniques in the literature have been proposed for the secure aggregation of genomic data. One of the very first work to demonstrate this proposed a technique to privately compute count queries on a genome dataset to ascertain which records meet a certain query predicate of genotypes and phenotypes [81]. This study used two third-party servers (one for data storage and the other to manage the cryptographic keys), and homomorphic encryption (HE) to allow data owners to encrypt their DNA data before uploading them to the data storage server. The limitations of this technique are two-fold: (1) computational cost incurred from encrypting the whole genomic database with HE, and (2) the query response time is slow due to the binary encoding scheme used to represent the DNA sequences and the query coupled

with multiple rounds of communication between the two third-party servers. In a bid to address limitation 1, a recent study proposed the use of a symmetric key encryption scheme (Advanced Encryption Standard in CTR mode) which is less costly than HE [82]. This study further optimized their solution to achieve less communication rounds by using only one third-party server with a secure cryptographic coprocessor (SCP) as opposed to the two servers used in [81]. Such a technique leverages the tamper-resistant feature of the SCP to address a malicious security model where the internal memory of the server is erased if tampering is detected. It is important to note that though an SCP addresses the security challenges in a realistic adversarial setting (malicious adversary), it is still limited in memory size and computation power. Another recent study [83] on addressing the computational cost issues of [81] proposed a trade-off between security and efficiency in that instead of encrypting the whole database to achieve stronger security as in [81], they partially encrypted the database while improving efficiency. Security and privacy of genome data in this model is achieved by partially encrypting the genome database, permuting genome sequences columns and inserting fake records to disassociate relationship between genomic sequence and sensitive attribute (e.g. disease). While it is vulnerable to Homer et al.'s attack [43], privacy is preserved under a semi-honest cloud model. Improving the query response time from limitation 2 has been proposed by recent studies [84, 85]. While [84] utilizes an efficient encoding technique for DNA sequences which makes it possible to represent a richer set of queries, [85] proposes a tree-based indexing technique for the efficient execution of queries. Furthermore, [85] guarantees data, query and output privacy as opposed to a previous studies [81, 82] in the same lines which only provided data and query privacy.

2.5.2 Secure GWAS and Statistical Analysis

GWAS is an approach used by researchers to evaluate the correlation between genetic regions (loci) such as SNPs and traits such as diseases between two sets of research groups - the case and control groups. This has, over the past few years, proven to be instrumental in enabling researchers to discern the genetic variations that drive

common and complex diseases such as diabetes and cancers, and in doing so, is laying the groundwork for personalized medicine. For this goal to be realized, large amounts of genomic data from different sources and studies need to be analyzed. This has so far been hindered by privacy issues related to sharing genomic data for GWAS. The genomic privacy research community has been actively investigating new ways to carry out GWAS securely while protecting the privacy of the study participants. Popular statistical methods such as Linkage Disequilibrium, Hardy-Weinberg Equilibrium, Cochran-Armitage Test for Trend and Fisher's Exact Test used in association tests have already been implemented securely [74].

The application of differential privacy for releasing GWAS statistics (MAFs, χ^2-statistics, p-values etc.) without compromising the privacy of participants especially the attack discovered by [43] is a well-known area in genomic privacy [7]. One of the key foci of work in this area is to find a better trade-off between data utility and the privacy of GWAS participants. A study in this direction proposed a relaxation to the adversarial settings of DP to achieve a higher utility while at the same time protecting membership disclosure [86]. Genotype and phenotype data used in GWAS, in most cases, are stored across multiple biobanks in different locations. Securing GWAS analysis in such distributed environments is vital in in-cooperating private data from different sources without privacy leaks. This was presented in a study where data (genotype and phenotype) is securely aggregated from multiple data collectors and secretly shared amongst computation servers [87]. Researchers can then submit analysis request to these servers where secure multi-party computation is used to process shares and output a result.

In GWAS, it is a common practice for researchers to perform meta-analysis which allows them to combine summary statistics from multiple studies and from different geo-locations to increase statistical power and reduce false-positives in their results. It is a well-established fact that these summary statistics are prone to inference attacks if not protected. This has been demonstrated in a recent study where researchers propose an SMC and differential privacy-based quality control procedures for meta-analysis [59]. The key focus of this work is to protect summary statistics while at

the same time securely checking for studies with quality issues in the meta-analysis pipeline. This protocol however suffers from the computation and communication overheads of SMC-based protocols. Apart from GWAS quality control procedures, another recent study proposed a privacy-preserving technique to correct population stratification (i.e. to account for false positive associations introduced by population structure) in GWAS by securely running PCA with SMC [88].

For large-scale GWAS of millions of individuals and about 10K SNPs, the secure evaluation of statistical analysis such as PCA using SMC based on homomorphic encryption requires over 30 years of computation time while the garbled-circuit-based approach incurs a communication overhead of roughly 190 PB, which is not feasible. A recent work [6] proposed a technique to reduce these overheads to a number of days for computation and 36 TB of data transfer. This technique uses SMC driven by secret sharing to facilitate GWAS quality control, population stratification analysis (based on PCA) and association tests while preserving the confidentially of participants.

Research on secure GWAS is still in its infant stage and further investigation with other cryptographic primitives and statistical methods such as logistic regression will be required for the quantification of privacy, test accuracy and efficiencies within various GWAS stages (i.e., pre and post-GWAS, quality control and population stratification). In particular, what trade-off between privacy and utility in differential privacy will be optimal for a given GWAS association test and what cryptosystems will give near accurate results as that obtained from carrying out GWAS in cleartext domain.

2.5.3 Secure Sequence Comparison and Matching

Sequence comparison is one of the most fundamental techniques to analyze DNA sequences for similarity or homology. This is often achieved by aligning sequences to evaluate the optimal cost of insertions, deletions and substitutions of nucleobases (A, C, G and T). Well-known sequence comparison methods such as dynamic programming methods (Smith-Waterman algorithm and Needleman-Wunsch), word methods (BLAST and FASTA) and their variants have been implemented securely by the cryp-

tographic research community to protect the privacy of DNA donors. An example of such work proposed an SMC-based technique for pairwise sequence comparison using dynamic programming (Smith-Waterman algorithm) [89, 90]. Both studies presented a setting where two parties can compute the edit distance between their sequences such that neither party learns anything about the private sequence of the other except the comparison result.

The use of edit distance for sequence comparison has seen applications in clinical setting such as similar patient querying (SPQ). This is useful where matching patients with similar genomic sequence helps inform a medical doctor of possible diagnoses and effective treatment options. Despite its usefulness, edit distance between two sequences is quadratic in time complexity. A recent study in this direction approximated the edit distance problem for querying similar patients [91]. Their approximation is based on partitioning sequences (database sequences and query sequence) into blocks according to a public reference sequence. Edit distance between these blocks are then precomputed and parties in the protocol apply GC to securely compute shares which are then summed locally to retrieve the approximate edit distance. This approximation method, however, leaks information about patients. Other related studies in the same line are [92, 93]. While [92] also suffers from an information leakage, [93] provides privacy guarantees for the genomic data, the query and the computation output.

Dynamic programming methods are computationally costly especially for a whole genome comparison. For privacy-preserving implementations, such overheads will always add up to the complexities inherent in the underlying cryptographic primitive, consequently rendering the application impractical. One way of dealing with such complexity is breaking the computation problem down so that more costly operations can be executed on high-end servers. This has been shown in a study where researchers proposed a dynamic programming technique in an SMC setting. The study decomposed the computation problem based on the sensitivity level of the genome data using program specialization by exploiting the fact that 99.5% of human genomes are similar and therefore not sensitive [64]. This allowed the data owner to perform

computation on the sensitive parts of the sequence while the third-party environment processes the rest of the sequence (non-sensitive nucleotides). Another approach to avoid the computation-intensive operations that come with dynamic programming is to opt for efficient comparison methods (e.g., BLAST and FASTA). One such study focused on securing BLAST algorithm by proposing a technique to securely outsource computation of DNA read-mapping to the cloud environment [71]. This study made use of secure hardware with field programmable gate arrays (FPGAs), on the cloud to seal computation and prevent sensitive information leakage to the rest of the server thereby protecting the DNA sequence.

The limitations of SMC and HE techniques such as scalability issues due to multiple communication rounds and introducing significant storage cost respectively are major roadblocks confronting the practical usability of secure sequence comparison frameworks in real-world clinical settings. Finding workarounds or solutions to these challenges together with the development of techniques to protect sequence search patterns will be a huge step. A recent study aimed at solving these challenges, proposed the use of private information retrieval (PIR) techniques with HE to run queries over encrypted genomic variants based on some comparison predicates [94]. This technique yields a faster query response time as opposed to implementation versions of sequence comparison solely based on HE. Another recent study went about the scalability issues discussed above by using a modified version of the predicate encryption (PE) scheme as opposed to SMC. This study addressed a sequence comparison privacy use-case in personalized medicine where genetic testing is used to diagnose and treat an individual [95].

The majority of privacy-preserving genomic techniques which involve two or more third-party servers assume a security requirement of non-colluding servers. This means third-party servers can collude to break the security of the system. A recent study sought to address this by proposing a collusion-resistant approach to securely outsource dynamic programming-based sequence comparison to a single cloud server thereby dismissing the non-colluding requirement [96]. The technique utilizes additive order preserving encryption (AOPE algorithm) which is homomorphic on addition and

preserves the numerical order of plaintext values thereby allowing the cloud to run comparison on encrypted genome sequences.

Though sequence comparison is amongst the widely covered areas in the implementation of privacy-preserving genomic techniques, most of them are built on SMC primitives which is complex in bandwidth. Other primitives, such as lattice-based or fully homomorphic encryptions to support arbitrary sequence comparison operations on ciphertexts are yet to be explored. Also, adversarial models based on assumptions of semi-honest and non-colluding parties should be adopted for a more practical and realistic setting of a malicious adversary.

Table 2.3: Security and efficiency comparison of secure genomic techniques

Category	**Scheme**	**ARC**	**CRYP**	**Security**					**Efficiency**	
				C	**I**	**QP**	**OP**	**AM**	**Comp Cplx**	**Comm Cplx**
Genomic aggregation	Kantarcioglu et al., 2008 [81]	SO	PE	●	○	○	●	SH	$O(nk)$PK + $O(n)$Q	$O(1)$
	Canim et al., 2012 [82]	SO	SCP, AES	●	●	○	●	MA	$O(n/b)$SE + $O(r)$Q	$O(1)$
	Ghasemi et al., 2017 [83]	SO	PE	●†	○	●	●	SH	–	–
	Nassar et al., 2017 [84]	SO	PE	●	○	○	●	SH	$O(nk)$PK + $O(r)$Q	$O(1)$
	Hasan et al., 2018 [85]	SO	PE,GC	●	○	●	●	SH	$O(nk)$PK + $O(nr)$BT + $O(\delta)$Q	$O(1)$
GWAS and statistical analysis	Kamm et al., 2013 [87]	SC	SMC,OT, SS	●	○	–	●	SH	$\gamma(O(n)$FC + $O(n)$TE)	$O(\gamma)$
	Tramer et al., 2015 [86]	–	DP	●	○	–	●	WA	–	–
	Huang et al., 2017 [59]	SO, SC	SMC,GC, DP	●	○	–	●	SH	$O(nlog^2n)$OS + $O(n)$OD	$O(n)$
	Sadat et al., 2018 [74]	SC	PE,SGX	●	○	○	●	SH	$O(nk)$PK + $O(n)$	$O(d)O(1)$
	Dan et al., 2018 [88]	SC	SMC,SS	●	○	–	●	SH	$O(n^2)$PCA	$O(n)$
	Cho et al., 2018 [6]	SC	SMC,SS	●	○	–	●	SH	$O(p+n)$PCA	$O(p+n)$
Sequence comparison	Atallah and Li, 2005 [89]	SO	SMC,OT, HE	●	○	–	●	SH	$O(n^2)$PK + $O(n^2)$CM	$O(\sigma n^2)$
	Jha et al., 2008 [90]	SC	SMC,OT, GC	●†	○	–	●†	SH	$O(n^2/\phi^2)$CM	$O(n^2/\phi^2)$
	Wang et al., 2009b [64]	SC	SMC,GC	●†	○	–	●	SH	$O(\rho n^3)$CM	$O(\rho n^3)$

Continued on next page

Table 2.3 Security and efficiency comparison of secure genomic techniques - Continued

	Xu et al., 2014 [71]	SO	AES, HMAC, FPGA	●	●	–	–	MA	$O(n/b)$SE + $O(1)$H + $O(n)$Q	$O(1)$
	Wang et al., 2017a [95]	SO	PRENC	●	○	●	●	SH	$O(n)$Enc + $O(n)$Q	$O(1)$
	Sousa et al., 2017 [94]	SO	SHE,AES, hash	●	○	●	●	SH	$O(n/b)$SE + $O(n)$H + $O(n)$Q	$O(1)$
	Asharov et al., 2017 [91]	SC	GC,OT, SS	○	○	●	–	SH	–	–
	Mahdi et al., 2018 [93]	SO	GC,AES	●	○	●	●	SH	$O(\Omega)SK$ + $O(nr)BT$ + $O(r)Q$	$O(1)$
	Wang and Zhang, 2018[96]	SO	AOPE, Hash	●	○	–	–	SH	$O(1)$Enc + $O(logv)$Dec + $O(n^2)$CM	$O(1)$
Genetic Testing	Troncoso-Pastoriza et al., 2007 [97]	SC	PE,SS, OT	●	○	●	●	SH	$O(nq\sigma)$PK + $O(n\sigma)$MP	$O(n(q+\sigma))$
	Bruekers et al., 2008 [66]	SC	SMC,hash, HE	●	○	–	●	SH	$O(nk)$PK + $O(n-t)$CM	$O(n^t)$IT; $O(n^{t+1})$ [PT,AT]
	McLaren et al., 2016 [98]	SO	PE,SMC, AES	●	○	●	●	SH	$O(n)$PK + $O(n/b)$SE + $O(n)$CM + $O(n)$MP	$O(m)$
	Jagadeesh et al., 2017 [79]	SO, SC	SMC,OT, GC	●	○	–	●†	SH	P1.,Max: $O(logp)$; P2.,Diff: $O(logp)$; P3.,CM: $O(n)$	P1.,Max: $O(logp)$ P2.,Diff: $O(logp)$ P3.,CM: $O(n)$
	Blanton et al., 2017 [65]	SC	SMC,Hash, OT	●	●	–	–	SH	$O(n)$CM + $O(n)$H	$O(n)$
	Chen et al., 2017 [73]	SO	SGX,AES-GCM, ECDSA	●	●	●	–	MA	$O(n/b)$SE + $O(r)$H + $O(1)$Q	$O(1)$

Continued on next page

Table 2.3 Security and efficiency comparison of secure genomic techniques - Continued

Notations:

AES: Advanced encryption standard; **AM**: Adversarial Model; **AOPE**: Additive order preserving encryption; **AT**: Ancestry test; **ARC**: Architecture; **BT**: Building tree operation; $\boldsymbol{b}$: Data block size used in AES; **C**: Confidentiality; **Comp**: Computation; **Comm**: Communication; **CRYP**: Cryptographic primitive; **Cplx**: Complexity; **CM**: Sequence comparison operation; **DP**: Differential privacy; **Dec**: Decryption function; $\boldsymbol{d}$: number of data owners; **ECDSA**: Elliptic Curve Digital Signature Algorithm; **Enc**: Encryption function; **FC**: Allele frequency counting; **FPGA**: Field-programmable gate array; **GC**: Garbled circuit; **H**: Hash operation; **HE**: Homomorphic encryption; **I**: Integrity; **IT**: Identity test; $\boldsymbol{m}$: # of iterations in an interactive protocol; **MA**: Malicious; **Max**: Maximum operation; **MP**: Muliplication operation; $\boldsymbol{n}$: Sequence length; **OT**: Oblivious transfer; **OS**: Oblivious sorting; **OD**: Oblivious de-duplication; **P1,2,3**: Protocol 1 ,2, 3; $\boldsymbol{p}$: # of study participants; **PE**: Paillier encryption; **PRENC**: Predicate encryption; **PCA**: Principal component analysis; **PK**: Public key encryption operation; **PT**: Paternity test; **Q**: Genomic query operation; $\boldsymbol{q}$: # of finite automata state; **QP**: Query Privacy; $\boldsymbol{r}$: # of records in genomic dataset; **SE**: Symmetric key encryption operation; **SM**: Semi-honest; **SMC**: Secure multi-party computation; **SHE**: Somewhat homomorphic encryption; **SS**: Secret sharing; **SO**: Secure outsourcing; **SC**: Secure collaboration; **TE**: GWAS test evaluation; $\boldsymbol{t}$: # of tolerable matching errors; $\boldsymbol{v}$: maximum plaintext value; **WA**: Weak adversary; $\boldsymbol{\delta}$: depth of index tree; $\boldsymbol{\Omega}$: # of tree nodes; $\boldsymbol{\gamma}$: # of secret shares

2.5.4 Secure Genetic/Genomic Testing

Genetic testing is the examination of variations in chromosomes, genes and proteins between an individual's genome and some biomarkers to determine certain disease risk, parentage, genealogy, etc. There are several genetic test types used for different purposes. These test are often performed on SNPs or the whole genome sequence (WGS). To the best of our knowledge, only disease susceptibility, identity, paternity, genealogical and compatibility tests have been implemented securely to preserve genome privacy.

One of the very first studies to secure a genetic test adapted finite automata to securely run DNA queries to determine an individual's disease risk [97]. The query is formulated as a regular expression and is run obliviously on the DNA sequence to check for disease markers. This study used secret sharing, homomorphic encryption

and oblivious transfer in an interactive protocol to ensure that both the query and the genome data were kept private. Most recent work has reduced the disease susceptibility testing and diagnosis problem to a secure two-party case where parties use interaction protocols to test for the presence of mutations and rare variants [98, 79]. The focus of [98] is to predict HIV-related cases, while [79] adopts frequency-based clinical genetics to propose three different protocols to diagnose patients with monogenic disorders.

In addition to securing the disease risk test, other studies have proposed cryptographic protocols for paternity, identity and ancestry tests. One such work presents a secure technique based on homomorphic Paillier encryption to match DNA profiles (short tandem repeats - STRs) from two parties to conduct paternity, identity and ancestry tests [66]. The secure matching problem is accomplished by constructing polynomials over the input STRs which are kept private and yields zero if there is a match. Another work proposes the use of garbled circuits in a two-party setting (SMC based) to securely evaluate (i) paternity tests over STRs from a parent and child, and (ii) genetic compatibility tests between partners to determine the risk of passing certain Mendelian-inherited diseases to their children [65]. Again, STRs are kept private from parties in the SMC protocol. However, the genetic compatibility test in this work leaks the disease which is being tested. This information leakage might be exploited by an adversary.

All the aforementioned secure genetic testing techniques are based on SMC, homomorphic encryption or a combination of both. SMC and homomorphic encryption-based techniques are known to have scalability issues due to overheads in storage, computation and communication, which makes them difficult to adopt for practical use over large scale genomic data or the full human genome [7]. A recent study proposed the use of secure cryptographic hardware (SCH) as an alternative to SMC and homomorphic encryption to implement a secure scalable practical genetic testing system [73]. This technique enabled the efficient and secure outsourcing of storage and genetic testing to an untrusted cloud environment for the purposes of disease diagnosis and personalized medicine. The secure hardware in the cloud is based on

Intel's Software Guard Extension (SGX) which provides a secure computation unit (enclave) where the genetic testing functions are executed. Despite the fact that SCH-based techniques are more scalable than SMC and homomorphic encryptions, they are limited in memory size (e.g., a single SGX machine is limited to 128 MB) which is crucial when processing large volumes of genomic data. The application of SGX to secure genomic computation is still new and further investigations will be required to establish its vulnerabilities, limitations and mitigation strategies for attacks including side-channel and in-memory attacks.

Though not directly related to genomic testing, the beacon service provides an opportunity for researchers to query a beacon network for the presence of an allele. This not only promotes data sharing but also facilitates research in areas of identifying disease markers for genomic testing. The re-identification attack by [40] on beacons as discussed in Section 2.2.1 has raised awareness on the need to protect beacon services. A recent study proposed two differentially private techniques, eliminating random positions and randomization of response, to protect a beacon [99]. Both methods are based on introducing inaccuracies to conceal the presence of an individual in a beacon database. Another study proposed a flipping strategy to optimize the trade-off between the utility and the privacy of the beacon service [100]. In this study, the discriminative power for each SNP in the database is calculated, and the top k SNPs with the greatest power are flipped. We, however, do not include these techniques in Table 2.2 as they are solely based on randomization and data perturbation techniques. The goal of Table 2.2 is to compare cryptographic techniques for protecting genomic data.

2.5.5 Comparison and Evaluation of Secure Genomic Techniques

Secure genomic computation techniques are evaluated using two criteria: (1) security guarantees achieved by the system as discussed in section 2.4.2 and (2) the efficiency of the technique in terms of computation and communication overheads. We compare the secure genomic techniques surveyed in this section using these criteria and present

them in Table 2.3 which we believe will give researchers an insight into the security goals and overheads incurred by the cryptosystems already proposed in the literature on specific genomic computation areas.

***Security*:** We evaluate and compare security guarantees in the table using symbols (i) ●: security guarantee is met as per definition in Section 2.4.2, (ii) ●†: security guarantee is met as per definition in Section 2.4.2, however with some information leakage. For instance, schemes with security bounded to the message (genome sequence) distribution or genome sequence length is leak during SMC protocol, (iii) ○: security guarantee not met, and (iv) –: security guarantee not available or not applicable to the framework as per the authors definition of their security goals. The prime security goal of any secure genomic technique is to provide confidentiality where a probabilistic polynomial time (PPT) adversary has negligible probability of breaking the scheme. This has so far been provided by all the schemes surveyed in this section. However, there are a number of ways an adversary can infer confidential genomic information without necessarily breaking the encryption system, for example, by observing the information leaked during computation or from the output, or using the attack techniques presented in Table 2.1. Incorporating integrity checks into secure systems is one approach to detect adversarial tampering aimed at learning sensitive information. Very few techniques in genomic privacy have adapted SCH which has tamper detection built into its coprocessors for integrity checking [82, 71, 73]. Other than SCH, coming up with genomic tamper resistant techniques has been challenging either because it is difficult to achieve or it incurs extra computation cost which might render the system impractical. This leaves an unexplored area in the genomic privacy domain to design techniques which provide a security guarantee of integrity.

***Efficiency*:** For fairness of the comparison between schemes, we evaluate efficiencies using asymptotic complexity (Big O notation) for (i) computation complexity as a function of the time it takes to encrypt/decrypt and perform the task such as sequence comparison and run queries on the genomic sequence, and (ii) communication complexity in terms of the number of computation rounds (iterations) between parties in the protocol. For instance, techniques based on SMC and OT involve multiple

interactions between parties in order to compute a function and as such, incur a communication complexity linear to the length of the genome sequence (i.e. $\propto n$). From Table 2.3, it is evident that secure hardware-based techniques [82, 71, 73] have the least computation cost which is linear to the genome sequence length (n) and communication in constant time of $O(1)$, whereas homomorphic encryption based techniques [89, 97] incur the most computation cost as a result of ciphertext expansions during encryption (i.e. encryption within a modular field with some exponentiation operation which is known to be more costly than other arithmetic operations– addition and multiplication) with security parameters chosen to provide stronger security requirements. In the case of an outsourced genomic storage model, this ciphertext expansion will increase data storage size and cost on third-party servers hosted by cloud providers (e.g., AWS, Google and Microsoft). It has been shown that privacy-preserving techniques for sequence comparison has scalability issues such as communication overhead which is linear to the sequence length (n) and number of parties in the protocol [89]. SMC techniques based on homomorphic encryption [89] incur excessive computation cost while those based on GC [90, 64] yield communication burdens. Loosely speaking, a better approach to yield efficiency in SMC would be to adopt secret sharing as applied in the work in [6].

It is important to note that the genomic computation which is been protected plays a vital role in estimating the efficiency of a secure framework. For example, the computation cost inherent to dynamic programming algorithms such as Smith-Waterman and Needleman-Wunsch for sequence comparison are quadratic ($< O(n^3)$) and as such when implemented in a secure domain will result in even higher overheads as is evident from the techniques [89, 90, 64] in Table 2.3.

2.6 Open Issues and Challenges

In this section, we summarize the open issues and challenges in designing techniques for securing genomic data and processing.

Gap between current approaches and their applicability: Work still needs to be done to transition techniques for secure genomic processing from theory to practice.

The majority of techniques proposed in literature are impractical for processing the human genome which is several sequences long. For the most part, it is the result of the underlying cryptographic system used. It is noteworthy to emphasize that there is no one cryptographic primitive which is best suited to protect genomic data for computation and storage. Any cryptosystem can be used. However, employing efficient versions of a cryptosystem and understanding whether it is applicable for a given genomic application area will be the first step towards achieving real-world feasible frameworks. For instance, SMC has been combined with differential privacy to solve privacy issues in GWAS [59], whereas homomorphic encryption and SMC have been used to complement each other to provide better efficiency and security guarantees in genetic testing and personalized medicine where biomarkers and test queries might be proprietary to test providers or pharmaceutical companies [98]. Despite these advances, more studies are still required to evaluate the performance of genomic applications in real-world privacy-preserving settings.

Tradeoff between security and efficiency: Security comes at the cost of efficiency. This means the more secure a scheme is, the lower its efficiency. Cryptographic systems have different ways of tuning encryption parameters for stronger security such as using large key sizes, large prime numbers in a modular ring, and increasing the difficulty of a mathematical problem (e.g., computing discrete logarithms). The more these techniques are tuned for stronger security guarantees, the more overheads they incur and the less efficient the framework becomes. Hence a secure genomic framework should be designed taking into consideration the tradeoff between the security required by the application and its efficiency.

Accuracy loss: There is always a price to pay for implementing privacy-preserving techniques. One of these is accuracy loss which is the margin of error between a secure implementation and its plaintext version. In genomic application processing, accuracy is critical and this error margin needs to be evaluated to estimate the impact it might have on clinical decisions for diagnostic testing or research results in GWAS. Current secure genomic techniques in the literature fail to address this; as a result, there is no measure of the impact they might have in a real clinical setting. Future

solutions should be designed to address accuracy with the goal of achieving negligible accuracy losses.

Personalized medicine and beyond: Advances in personalized medicine are beginning to use techniques such as artificial intelligence (AI) to classify mutations that contribute to diseases such as tumor growth, for example, classifying mutations in the BRCA1 and BRCA2 gene for breast cancer. These techniques are instrumental in the realization of personalized medicine and the transformation of health care delivery, and they are already being applied by companies in the industry such as Deep Genomics [101] and Atomwise [102]. However, protecting the privacy of patients remains the most critical issue with the commercialization of personalized medicine. Thus, designing secure AI models for personalized medicine while preserving the privacy of patients will be an interesting direction to explore.

2.7 Conclusion

The advancement of genome sequencing techniques is driving the ease of access and the collection of genomic data for storage, sharing and processing. This comes with increasing security and privacy concerns which are yet to be sufficiently addressed for purposes such as health care delivery, research and direct-to-consumer services. In this chapter, we explored and surveyed the relevant work on privacy attacks and privacy-preserving techniques for genomic data. An adversary is always exploring new ways to compromise the privacy of an individual via his/her DNA sample for personal gain, blackmail or some other dubious reasons. As a result, further research is required to propose techniques to thwart these attacks not just in theory but in the practical application to real-life settings as well. We believe the comparison of secure genomic techniques based on computation and communication overheads and the future directions provided in this chapter will serve as a guide for the genomic privacy researcher to achieve this goal.

Chapter 3: Privacy-based Drug Dosage Prediction in Genotype and Clinical Personalized Medicine

Continuous improvement in biomedical research is evolving medicine from the traditional practice of "one-size-fits-all" to a more personalized medicine approach where health care is tailored on an individual basis. This makes it possible for more accurate diagnoses and safer drug prescriptions. In the field of personalized medicine, dosing models are being developed to predict drug dosage based on patients genotype. As the human genome is sensitive, current dosing models are challenged by security issues which might lead to breaching patients privacy. In this chapter, we develop novel secure techniques using warfarin regression models, homomorphic and secure two-party computation (SMC) protocols to predict warfarin dosages specific to patients genetic variants while protecting their privacy. To the best of our knowledge, our work is the first to develop a secure genotype-guided dosage prediction framework. To improve the performance of our scheme, we develop a new SMC comparison protocol to securely compare patients single nucleotide polymorphism (SNP) states using homomorphic and blinding techniques as opposed to bit decomposition which is computationally expensive. In addition, we propose a new encoding method for patients genetic variants, race and age which avoids the high computation cost that comes with secure string matching operations. We run our experiments on a dataset of real patients who are commencing warfarin therapy for thromboembolic disorder and our results reveal that (i) our proposed scheme is secure and efficient for real-world clinical settings, and (ii) our secure estimated warfarin dosages are similar to

NOTE: The content of this chapter has been submitted to *ACM Transactions on Privacy and Security.*

Mohammed Yakubu, A., & Chen, Y. P. P. Privacy-based Drug Dosage Prediction in Genotype and Clinical Personalized Medicine. *ACM Transactions on Privacy and Security.* (Under Review).

that of ground truth dosages with negligible losses.

3.1 Introduction

Over the past decade, advances in the overlapping fields of genomics, computer science, medicine and health care is bringing to light how medical conditions, diseases and drug responses in each person are influenced by genetics. As a result, this is transforming health care delivery from the traditional practice of "one-size-fits-all" which is based on population averages to a more individual tailored approach [103]. The traditional approach of health care delivery uses clinical symptoms and a few classic laboratory markers to give the same treatment for all patients with similar diseases and symptoms. The result of this is that due to the genetic heterogeneity amongst patients, drugs and treatment work for some patients but might lead to adverse drug reaction [104] in others. Personalized medicine which tailors health-care delivery based on patients unique genetic makeup aims to address the limitations of the traditional approach by combining pharmacology and genomics to develop pharmacogenetic dosing models to guide safe and effective drug medications tailored to the variabilities in the human genome [105]. The initial emphasis of pharmacogenetic guided dosing models are on drugs with narrow therapeutic windows i.e. the range of dosages that produces the greatest therapeutic benefit in patients without adverse side-effects. Chief amongst these drugs is warfarin which has a narrow therapeutic window with lots of interindividual variability in response to treatment. Warfarin, an oral drug, is the most commonly prescribed medication for the treatment of thromboembolic disorders and due to its interindividual variability in therapeutic response, it might lead to bleeding complications in patients if not prescribed based on the genetic reaction to the drug [106]. In addition, the U.S. Food and Drug Administration (U.S. FDA) have approved and labelled genotypes responsible for metabolizing warfarin as actionable pharmacogenomic biomarkers to aid in warfarin dosing [107].

To address the above problem, researchers in the field of pharmacogenetics are developing personalized warfarin dosage algorithms which computes patients warfarin dosages based on their clinical and genomic data [108]. However this comes with

serious security and privacy challenges as patients genomic data are required to be protected in accordance with regulations such as Health Insurance Portability and Accountability Act (HIPAA), and more importantly, the genomic data used in the dosage computation model is highly sensitive. This is because the human genome is unique and encodes information about a person such as his predisposition to diseases. In an event where patients genotype is compromised by an adversary, it might lead to privacy breach cases such as genetic discrimination in health or life insurance [109]. In addition, medical and health facilities are constrained in computing resources to process health informatics applications such as drug prescription models and disease diagnosis, so they outsource storage and computation to third-party environments, such as the cloud. This consequently worsens the security and privacy situation as third parties are semi-trusted. A common practice in such a case is to encrypt the genomic data with standard encryption techniques such as advanced encryption standard (AES). However, this approach does not allow computation to be performed on the encrypted data. Thus, there is the need for patients genomic data to be secured while allowing the computation of pharmacogenetic guided warfarin dosing.

As a solution to the above issue, we propose novel secure techniques for the personalized prescription of warfarin's dosages. Our proposed scheme is built on top of pharmacogenetic guided warfarin dosage prediction models [108] and makes use of homomorphic encryption and secure multiparty computation (SMC) protocols between two semi-honest servers to securely carry out operations which are not feasible solely based on homomorphic computation. Also, our proposed scheme follows an outsourced model where the goal is to offload storage and computation from resource constrained medical facilities to the cloud. In previous studies, several techniques have been proposed to protect genomic data for storage and computation. These techniques can be categorized into (i) secure protocols for GWAS and statistical analysis [6, 77], (ii) secure techniques for DNA sequence comparison and matching [93, 110], and (iii) secure protocols for genetic testing for disease susceptibility and rare variants [73, 111]. Despite these advances, protecting the privacy of patients for applications in personalized medicine has not been explored in detail and there is more work to

be done in securing dosage prediction. To the best of our knowledge, this is the first work to provide a secure genotype-guided dosage prediction framework. However, it is noteworthy to mention the recent work in [98] also in the lines of genotype-guide medicine where a DNA-based prediction model is used to diagnose HIV-related cases.

Personalized medicine is an emerging area and it is still challenging to preserve the privacy of patients in this domain for computations of arbitrary operations. In an ideal case, fully homomorphic encryption scheme (FHE) can be used to allow an unbounded number of operations on the encrypted data; however, current implementations are still not efficient. To improve the efficiency of FHE, leveled-FHE and their variants such as ring learning with errors (ring-LWE) have been introduced to evaluate functions of a bounded depth [112]. Despite the recent advances made by leveled-FHE, there are still scalability and efficiency issues especially for comparison operations between two ciphertext [113]. Our solution solves this scalability and efficiency issues by combining partial homomorphic encryption (modified Paillier cryptosystem) with SMC to demonstrate a proof-of-concept implementation of novel secure protocols for comparison, multiplication and warfarin dosage computation in a secure outsourced model. In summary the contributions of this chapter are as follows:

- To the best of our knowledge, this is the first work to design a new secure personalized dosage prediction framework where a regression model is used to predict patients warfarin dosages, based on their genomic and clinical data.

- We develop a new set of novel one-round homomorphic based SMC protocols between two cloud servers. We categorize these protocols into (i) sub-protocols to perform operations on ciphertext, and (ii) main protocols to invoke the sub-protocols to securely compute patients warfarin loading and maintenance dosages.

- We propose new modifications to warfarin dosage regression models [103, 114, 115] to make them work in secure domain. Specifically, we propose new secure functions to compute patients (i) maintenance dosage based on the summation of scaled genomic and clinical variables, and (ii) loading dosage where we propose a secure function to estimate the half-life of warfarin.

- Recent secure integer comparison solutions either incur high computation or communication cost due to bit decomposition and multiple communication rounds [116, 113]. We propose a new SMC based comparison protocol to securely compare SNP states using homomorphic and blinding techniques. We improve the performance of our protocol by avoiding bit decomposition.

- We demonstrate through extensive experiments on a real-world patients dataset that our proposed scheme is efficient with negligible accuracy losses and provides the security requirement of patients data confidentiality and computed dosage privacy. Furthermore, it is efficient in computation and communication, and practically feasible in a real-world clinical setting.

The remainder of this chapter is organized as follows. In Section 3.2, we discuss the preliminaries and background required to understand our proposed scheme. We formalize our system model and security requirements in Section 3.3. Section 3.4 presents the details of our proposed framework. We then evaluate the security and experimental analysis of our proposed scheme in Sections 3.5 and 3.6 respectively. Finally, Section 3.7 concludes this chapter.

3.2 Preliminaries and Background

In this section, we discuss the pharmacogenetic basis of warfarin and the background on homomorphic encryption (i.e. Paillier cryptosystem) and multiparty computation which serves as building blocks for our proposed framework.

3.2.1 Pharmacogenetics of Warfarin

Warfarin is the most commonly prescribed oral anticoagulant drug for the treatment of thromboembolic disorders (i.e. treatment of blood clots that might cause stroke, heart conditions etc.). Warfarin goes through an anticoagulation process (as shown in Figure 3.1) to thin blood. It has a narrow therapeutic range[1] with a wide interindividual variability which makes it difficult to dose [108]. Inappropriate dosing might lead

[1]The range of concentrations at which a drug is expected to achieve the desired therapeutic effects with minimal toxicity [117]

to (i) blood clots in some patients putting them at risk of having a stroke, or (ii) bleeding in others. These complications are the most common reasons for emergency room visits reported to the U.S. FDA [108]. In the rest of this section, we introduce the gene polymorphisms that influence warfarin dosing and refer the reader to [5] for the details of warfarin's pharmacogenetics.

Genome-wide association studies (GWAS) is an approach used by researchers to identity common genetic variants or mutations (usually single nucleotide polymorphisms) that are statistically associated with a specific trait or disease [118]. GWAS has been performed on the pharmacogenetics of warfarin and has established that its interindividual variability is a result of 41% genetic factors, 10% non-genetic factors and 49% unknown [5]. While the mutations are public knowledge, patients still prefer to keep them confidential to protect kin privacy. This is because an adversary with access to the patients DNA mutations can apply completion attack techniques to reconstruct DNA sequences of the patient's family members [119]. We show the contributing factors that has been discovered by GWAS for metabolizing warfarin in Figure 3.1. The 41% genetic contribution corroborates the fact that for efficacy, the polymorphisms of genetic variants responsible for metabolizing warfarin should be used to guide warfarin dosing. The genes with the strongest correlation that have been incorporated into warfarin dosing algorithms are CYP2C9 and VKORC1. The CYP2C9 genotype is encoded using the star allele nomenclature and its reference allele is CYP2C9**1*. Individuals homozygous for CYP2C9**1* (i.e. CYP2C9**1*1*) have the normal metabolizer phenotype. The most common carrier alleles of CYP2C9 responsible for a reduction in warfarin metabolism are CYP2C9**2* and CYP2C9**3*. Patients with one or two copies of CYP2C9**2* or **3* are at greater risk of bleeding during warfarin therapy [108]. The VKORC1 genotype has rs9923231(G>A) as the most common SNP significantly associated with warfarin sensitivity [120]. Patients with one or two copies of rs9923231(A) require lower warfarin dosages than those homozygous for rs9923231(G). We present in Table 3.1 the polymorphisms of VKORC1 and CYP2C9, and their impact on warfarin therapy.

Anticoagulation therapy with warfarin usually starts with a loading dosage fol-

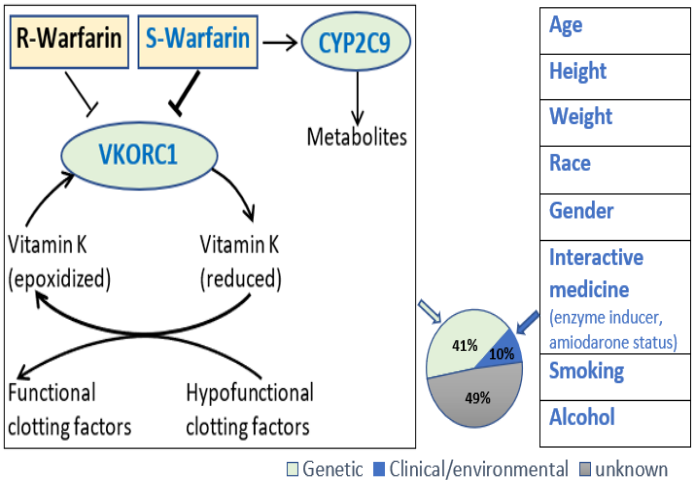


Figure 3.1: Contributing factors to warfarin response. (**Left**) Genetic factors (41%) showing the interaction between CYP2C9, VKORC1 and warfarin anticoagulation cycle. (**Middle**) Pie chart showing factors in percentage that influences warfarin's interindividual variability. (**Right**) Clinical factors (10%) which has been identified to influence warfarin therapy.

lowed by a maintenance dosage. Pharmacogenetic-based algorithms for warfarin loading and maintenance dosages have been developed from regression models by combining the SNPs of CYP2C9 and VKORC1 discussed previously, non-genetic data such as age, height etc., and gene-drug interaction [108, 121]. Each SNP of CYP2C9 and VKORC1 has a contributing weight from GWAS which is used in these predictive algorithms. A good example is the warfarin dosing portal [115] which is supported by national institutes of health (NIH). Recently, the U.S. FDA have recognized and labelled CYP2C9 and VKORC1 as actionable pharmacogenomic biomarkers of warfarin response [107]. As a result, they have approved some in vitro companion diagnostic genetic test kits to genotype CYP2C9 and VKORC1 to aid with warfarin dosing.

Table 3.1: The influence of VKORC1 and CYP2C9 on warfarin therapy with respect to enzyme activity

Gene	Polymorphisms	N or C	EA	Dose recommendation
VKORC1	rs9923231*(GG)*	N	LWS	ND
	rs9923231*(AG)*	C	MWS	LWD relative to GG
	rs9923231*(AA)*	C	HWS	LWD relative to GG
CYP2C9	**1*1*	N	NM	ND
	**1*2*	C	RM	LWD
	**1*3*	C	RM	LWD
	**2*2*	C	RM	LWD
	**2*3*	C	RM	LWD
	**3*3*	C	RM	LWD

Notations: **C**: Carrier; **EA**: Enzyme activity; **N**: Normal; **NM**: Normal metabolizer; **RM**: Reduced metabolizer; **ND**: Normal dosage; **LWD**: Lower dosage; **HWS**: High warfarin sensitivity; **MWS**: Moderate warfarin sensitivity; **LWS**: Low warfarin sensitivity

Loading dosage

The loading dosage is an initial higher dosage administered over the first three days (i.e. loading dosage day 1, 2 and 3) of commencing warfarin therapy to achieve a rapid therapeutic response [121]. This ensures that patients reach the therapeutic range quickly without overshooting their target international normalized ratio (INR) of 2 to 3. The pharmacogenetic-based loading dosage [108] is estimated from patients genetic polymorphism of the CYP2C9 genotype and the half-life of warfarin i.e. the time it takes for the concentration of warfarin in the body to be reduced by 50%. Below equation 3.1 shows the most relevant model [114, 115] for estimating patients loading dosage over the first three days where k is the warfarin elimination rate which is dependent on the CYP2C9 genotype, t is the elimination time interval in hours and $\mathcal{MD}$ is the estimated maintenance dosage. We call this model the baseline model for estimating loading dosage (**BL**).

$$\mathcal{LD}_{1,2,3} = \mathcal{MD} \times \frac{\left((1 - e^{-kt})^{-1} - (1 + e^{-kt} + e^{-2kt})\right)}{\left(1/3 + 2e^{-kt}/3 + e^{-2kt}\right)} \tag{3.1}$$

Maintenance dosage

After reaching the therapeutic range with the loading dosage, patients are then put on a maintenance dosage which is equal to the rate of elimination of warfarin at a steady state [122]. This ensures the concentration of warfarin in the body is at the appropriate level during therapy. The pharmacogenetic-based maintenance dosage [108] is derived from patients (i) clinical data: age, height, weight, race, enzyme inducer status, amiodarone status, and (ii) genetic data: polymorphisms of CYP2C9 and VKORC1 genotype. The most relevant model [114, 115] (i.e. the IWPC algorithm) for estimating patients weekly warfarin maintenance dosage is given as

$$\mathcal{MD} = \big(c + w_a\texttt{Age} + w_h\texttt{height} + w_w\texttt{weight} + w_{r_1}\texttt{rs9923231}\textit{(AG)} + w_{r_2}\texttt{rs9923231}\textit{(AA)} + w_{c_1}\texttt{CYP2C9}\textit{*1*2} + w_{c_2}\texttt{CYP2C9}\textit{*1*3} + w_{c_3}\texttt{CYP2C9}\textit{*2*2} + w_{c_4}\texttt{CYP2C9}\textit{*2*3} + w_{c_5}\texttt{CYP2C9}\textit{*3*3} + w_{c_d}\texttt{amiodarone}\big)^2 \tag{3.2}$$

where c is a constant coefficient, $w_a, w_h, w_w, w_{r_1}, w_{r_2}, w_{c_1}, w_{c_2}, w_{c_3}, w_{c_4}, w_{c_5}$ and w_{c_d} are coefficients for their respective patients clinical and genotype data. This is the baseline model for estimating patients warfarin maintenance dosage which we denote as **BM**. The problem with the baseline models **BL** and **BM** is that they expose patients sensitive clinical and genomic data in the clear. Thereby putting patients security and privacy at risk. The genomic privacy risk associated with exposing the patients sensitive clinical and genomic data are (i) attribute disclosure attacks: this privacy attack occurs when an adversary uses the exposed genomic data to predict sensitive attributes of the patient, such as phenotypes, disease association and drug abuse [17, 47] (ii) completion attacks: this occurs when an adversary uses genotype imputation techniques to reconstruct the patient's genetic information from partially compromised DNA data or a family member's DNA sequence [17, 14]. In our proposed framework we decompose and reformulate **BL** and **BM** by developing new secure SMC techniques to ensure patients security and privacy.

3.2.2 Paillier Cryptosystem

The Paillier cryptosystem is a public key cryptosystem with semantic security and additive homomorphic properties. In our proposed scheme, we use a variant [123, 98] of this cryptosystem to encrypt patients data. This variant provides proxy re-encryption which allows one party to alter a ciphertext so that it can be decrypted by another party without revealing his private key and consists of the following algorithms.

Key generation

Let $n = pq$ for two safe primes p, q of bitlength k, and g be a generator of order $(p-1)(q-1)/2$ which is computed as $g = -a^{2n}$ (a is a random number from $a \in \mathbb{Z}^*_{n^2}$);

then the public key is $pk = (n, g, h = g^{sk})$, and the private key is $sk \in [1, n^2/2]$. Furthermore, the private key is randomly split into two shares sk_1, sk_2 i.e. $sk_1 \in [1, n^2/2]$ and $sk_2 = sk - sk_1$.

Encryption

Given a message $m \in \mathbb{Z}_n$, The encryption function $E(.)$ converts it to a ciphertext pair $[m] = (c_1, c_2)$; where $c_1 = g^r \bmod n^2$ and $c_2 = h^r(1 + mn) \bmod n^2$. The encryption is given as $E(m) = (c_1, c_2)$.

Partial Decryption

The ciphertext $[m]$ can be partially decrypted with a share of sk_i $(i \in \{1, 2\})$ as follows: $PD_{sk_i}([m]) = [m]' = (c_1', c_2')$; where $c_1' = c_1 \mathrm{mod}\, n^2$, and $c_2' = c_2 c_1^{-sk_i} \mathrm{mod}\, n^2$.

Decryption

Let $D(.)$ be a decryption function, then to decrypt an encryption $[m] = (c_1, c_2)$ of the message m is computed as follows: $D([m]) = L(c_1 c_2^{-sk} \mathrm{mod}\, n^2) = m$; where $L(u) = \frac{u-1}{n}, \forall u \in \{u < n^2 | u = 1 \mathrm{mod}\, n\}$.

Cipher Refresh

The ciphertext $[m]$ can be re-encrypted without changing the original message m by randomly choosing $r \in [1, n/4]$ and calculating $CR([m]) = \hat{[m]} = (\hat{c_1}, \hat{c_2})$; where $\hat{c_1} = g^r c_1 \mathrm{mod}\, n^2$, and $\hat{c_2} = h^r c_2 \mathrm{mod}\, n^2$.

Additive homomorphic properties

The Paillier cryptosystem is additive homomorphic which allows certain operations to be performed on ciphertext. Suppose the messages $[m_1]$ and $[m_2]$ are ciphertexts under the same public key pk then the following homomorphic properties hold:

- *Addition*: The product of the ciphertexts $[m_1]$ and $[m_2]$ corresponds to the sum of their plaintext m_1 and m_2. This is given as $D([m_1] \times [m_2]) = m_1 + m_2$.

• *Scalar multiplication*: The ciphertext $[m_1]$ raise to the power a constant k which corresponds to the product of the constant k and the plaintext m_1. This is computed as $D([m_1]^k) = k \times m_1$.

It is important to note that the Paillier cryptosystem does not support multiplication and division in the ciphertext domain. We do this with SMC-based protocols which we present in the coming section. For brevity, in the rest of this chapter, we represent the encryption of a message m by $[m]$, its refreshed cipher with $[\hat{m}]$ and its partial decryption with $[m]'$.

3.2.3 Multiparty Computation (MPC)

Multiparty computation enables a group of parties $p_1, p_2, ..., p_m$ to jointly compute a function over their private data $d_1, d_2, ..., d_m$ without disclosing any parties sensitive data except for the computation result $f(d_1, d_2, ..., d_m)$ [124]. In this work we implement a secure two-party computation (SMC) case of MPC based on homomorphic encryption to securely compute patients data. In other words, we perform computation such as division and comparison which are not feasible solely based on homomorphic encryption via SMC between two parties i.e. the cloud platform (CP) and the computation service provider (CSP). Current efficient implementations of MPC are based on (i) secret sharing (SS), and (ii) homomorphic encryption (HE). Our scheme follows a secure outsourced model [125] where storage and computation is offloaded to the cloud. To realize this, we need to find a balance between computation, communication, and storage cost as cloud resources are on a pay-per-use basis. In an SS based SMC scheme, shares of SNPs are created and distributed to multiple CSPs. Each CSP stores a copy of the share for collaborative computation, thereby resulting in (i) increased communication cost [126], (ii) increased storage overhead, and (iii) increased processing power as all shares have to be computed and updated for each operation [126]. Based on these overheads, we choose HE based SMC to achieve the computation, communication, and storage tradeoff using a non-colluding two-cloud server setup. It is important to note that such setups have recently been utilized to securely outsource computation models with good efficiency to achieve the

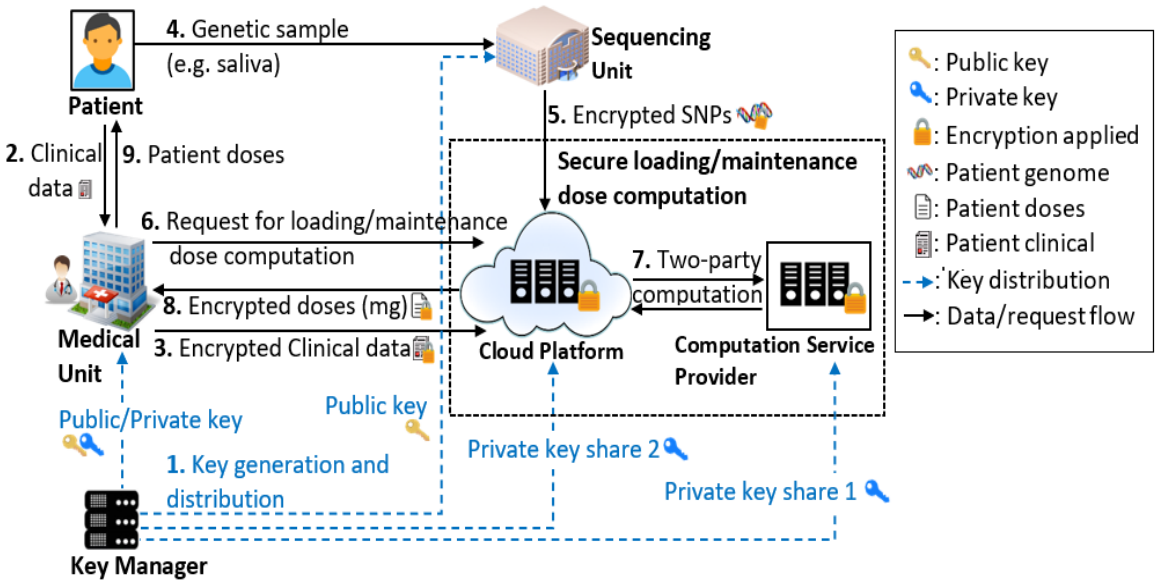


Figure 3.2: The system model for our secure pharmacogenetic warfarin dosage prediction scheme. The key manager (KM) initializes the system by generating the private key sk, public pk, and random shares (sk_1, sk_2) of sk. KM then securely transmits pk and sk to the medical unit (MU), pk to the sequencing unit (SU), sk_1 to CP and sk_2 to CSP. The MU and SU then acquire patients clinical and genomic data respectively. The MU and SU preprocess and encrypt patients clinical and genomic data respectively and upload them to CP for storage and computation. Now the MU wants to obtain patients maintenance and loading dosage and sends a request to the CP. The CP then collaborates with CSP to securely compute the dosages and returns the results to MU. Finally, MU decrypts and postprocesses the results to obtain patient dosages.

computation, communication, and storage tradeoff as opposed to using SS based SMC [98]. This does not mean SS based SMC do not perform well in other security setups. For instance, in a secure collaboration setting it has been applied in deep learning to securely train a predictive model for drug discovery [126].

3.3 System Model and Security Requirement

In this section, we briefly discuss the functions of the various entities in our system model, and outline the security requirement and threat model.

3.3.1 System Model

Our proposed scheme as shown in Figure 3.2 considers a system model where a resource constraint medical unit outsources patients data (clinical and genomic) to the cloud for storage and processing. As depicted in Figure 3.2, the system model com-

prises the patient, medical unit (MU), sequencing unit (SU), key manager (KM), cloud platform (CP) and computation service provider (CSP).

- Patient: We consider patients who are about to commence anticoagulation therapy with warfarin medication. Patients give their clinical and genomic data to assist the MU to advise and determine a stable dosage.
- KM: KM is a trusted entity whose responsibility is to create and distribute private and public keys for encryption and decryption respectively.
- MU: Generally MU provides medical services to patients. It uses its public key to encrypt patients clinical data.
- SU: The SU sequences patients genotype for CYP2C9 and VKORC1. The SU then encrypts the resulting SNPs with its public key and uploads them to the CP.
- CP: The CP provides storage and computation services to MU. It stores and computes on encrypted patients clinical and genomic data.
- CSP: The CSP collaborates with CP to provide online computation services in a secure two-party manner.

3.3.2 Security Requirements

Our proposed scheme addresses a security requirement for an outsourced model under a non-colluding assumption between two-cloud servers where storage and processing of genomic data is offloaded to a semi-honest cloud environment. The following is our security requirements.

- Data confidentiality: Sensitive patients genomic and clinical data should not be revealed to unauthorized entities such as CP and CSP at storage and during computation.
- Output privacy: The computed loading and maintenance dosages should not be leaked to any party other than the intended recipients (i.e. the patient and the MU).

In addition to these security requirements, we assume that KM is trusted entity i.e. KM honestly generates and securely distributes private and public keys to various entities in the framework. Furthermore, we assume the parties MU, SU, CP and CSP are semi-honest. This means these parties faithfully follow the protocol to securely compute patients dosages, but might try to learn sensitive information from the data they receive and store. Finally, we assume that CP and CSP are non-colluding, meaning that they do not come together to learn patients sensitive data. In practice, the two non-colluding cloud server setup have been used in a real-world setting where each server belongs to a different cloud service provider such as Microsoft Azure and Amazon EC2 [127]. The idea behind this is that the cloud providers are competitors and economically driven with the commercial interest of non-collusion.

3.4 Secure Pharmacogenetic Warfarin Dosage Prediction

In this section, we discuss the workflow of our proposed scheme for secure dosage prediction. We start by giving an overview of our scheme. Let $\mathcal{Q}$ denote a set of n patients data $\mathcal{Q} = \{\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_n\}$ where each patient data consist of clinical data $\beta = \{$age, height, weight, race, enzyme inducer status, amiodarone status$\}$ and genomic data SNPs $= \{$VKORC1-$rs1057910, CYP2C9\}$, i.e. $\mathcal{Q}_i = \{\beta, \text{SNPs}\}$ for all $i \in \{1, 2, \ldots, n\}$. We summarize the steps and workflow of activities involved in our scheme, as shown in Figure 3.3.

Step 1. **Initialization**: As shown in Figure 3.2 and 3.3, KM initializes the system by generating the private key sk and the public pk, and randomly creating two shares (sk_1, sk_2) of sk. KM then securely transmits pk, sk to MU, pk to SI, sk_1 to CP and sk_2 to CSP.

Step 2. **Preprocessing and encryption**: Next is the data encryption stage where MU and SU first preprocess (scaling and encoding) β and SNPs respectively, then encrypt them with pk, and upload them to CP for storage and computation.

Step 3. **Secure computation**: Upon receiving the maintenance and loading dosage computation request from MU, CP performs the computation on encrypted data with

some rounds of online secure collaboration with CSP to execute functions and sub-protocols.

Step 4. **Decryption and postprocessing**: Finally, CP sends the encrypted results to MU. MU then decrypts and postprocesses the results to obtain the actual dosages.

3.4.1 Preprocessing

In our framework, patients genomic and clinical data are preprocessed prior to encryption by MU and SU. The goal of our preprocessing is to scale real-valued ($\mathbb{R}$) patients height and weight to integer ($\mathbb{Z}$), and to encode patients SNPs, race and age.

(1) Scaling patients data: Most cryptosystems such as the Paillier cryptosystem operate in a ring modular of some prime integer number (i.e. within an integer domain $\mathbb{Z}$) and as such, cannot perform operations on real numbers. As a result, patients data such as height and weight which are measured as real numbers cannot be processed by these cryptosystems. To address this issue, we convert real-valued patients height and weight to integers. We do this by multiplying them with a constant scale factor 10^d, where d is some integer value. This is given by: $\beta_i' = (\beta_i + \mathcal{E}_r) \times 10^d$; where i belongs to a set of indexes for patients height and body weight, d is some positive integer value, and $\mathcal{E}_r$ is the rounding error (data losses). We approximate $\mathcal{E}_r$ as in [128] and bound it by:

$$\mathcal{E}_r \leq |1/2 \times 10^{-d}| \tag{3.3}$$

We apply the same scaling technique to convert the weights of β and SNPs from real values to integers which we discuss in the coming section.

Remark 1 *We acknowledge that converting patients height, body weight and variable (β, SNPs) weights from real to integer numbers leads to some data loss as a result of the rounding error $\mathcal{E}_r$. However, these losses are negligible and the resulting dosage predicted by our scheme converted to the nearest* $0.5mg$ *used by real-life warfarin prescription conversion charts is the same as the computation results from the ground truth (i.e. computation on plaintext). We present a formal proof in the coming section*

to justify this claim.

Table 3.2: Encoding for patients data (race, VKORC1-rs9923231 and CYP2C9 polymorphisms)

Data				**Encoding**
Race (race is encoded as enumerated type)				
White European				1
Black/African American				2
Asian				3
Missing or mixed				0
VKORC1-rs9923231 polymorphisms (major allele=G; minor allele=A. We propose the use of VKORC1's MIA as encoding as highlighted below)				
	MJA	MIA		
GG	2	0		0
AG	1	1	→	1
AA	0	2		2
CYP2C9 polymorphisms(major allele=**1*; minor allele=**2, *3*. We propose CYP2C9's encoding as the sum of its MJS as highlighted below)				
	MJS	MIS	Sum of MIS	
**1*1*	1, 1	0	0	0
**1*2*	1	2	2	2
**1*3*	1	3	3	3
**2*2*	0	2, 2	4	4
**2*3*	0	2, 3	5	5
**3*3*	0	3, 3	6	6
Notations: **MJA**: Major allele count; **MIA**: Minor allele count; **MJS**: Major allele star values; **MIS**: Minor allele star values				

(2) Encoding patients data: In our proposed secure scheme, we encode patients SNPs (i.e. CYP2C9 and VKORC1 polymorphisms), race and age with integer values. We do this to avoid string matching operations. Despite the fact that matching strings is feasible in secure domain, it is slow and comes with high computation cost. We encode patients race as an enumerated type as proposed in Table 3.2. However, for VKORC1, we encode the polymorphism rs9923231(G>A) as the count of the minor allele; where G is the major allele and A is the minor allele at SNP position rs9923231. For example, we encode a patient with SNP rs9923231(GG) as 0, rs9923231(AG) as 1 and rs9923231(AA) as 2. For CYP2C9 with a major allele of **1* and minor alleles of **2* and **3*, we use the sum of the minor allele star values as encoding. For instance a patient with CYP2C9**1*1* will be encoded as 0 (since there is no minor allele), CYP2C9**1*2* will be encoded as 2, CYP2C9**2*2* encoded as 4 (i.e. sum of minor allele star values $2 + 2 = 4$) and so on. Table 3.2 lists the data (i.e. race, CYP2C9

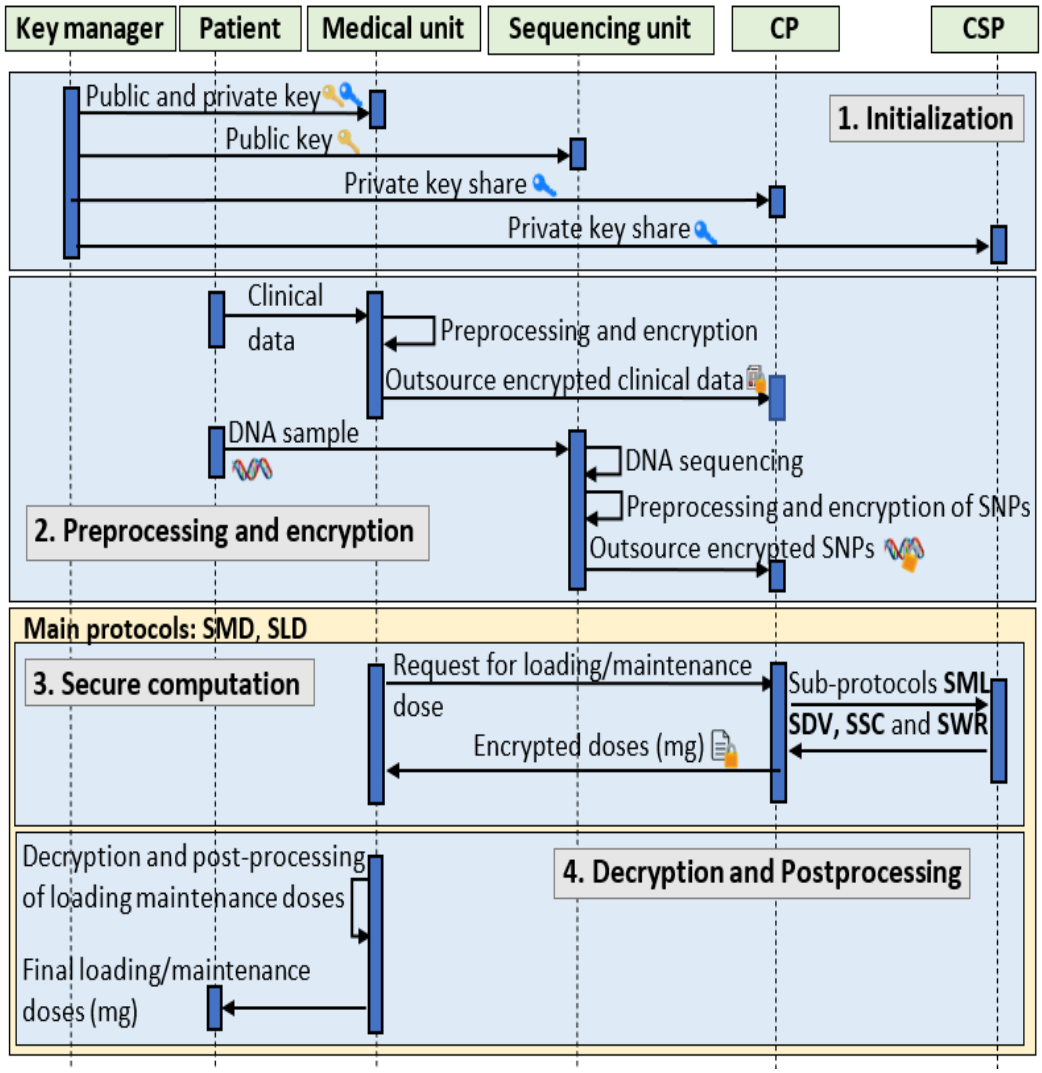


Figure 3.3: The workflow of activities in our proposed secure pharmacogenetic warfarin dosage prediction scheme. This consist of the initialization, preprocessing and encryption, secure computation, and decryption and postprocessing steps.

and VKORC1) and the proposed encoded values used in our scheme. However, for patients age, we use the same range encoding present in the original IWPC algorithm (we refer readers to [108] for details). In the rest of this chapter, we represent the encoded SNPs as $S = \{S_v, S_c\}$, where S_v and S_c are encodings for VKORC1 and CYP2C9 polymorphisms respectively, and encoded race as S_r.

3.4.2 Secure Protocols

Our proposed secure protocols for warfarin maintenance and loading dosage computation as shown in Table 3.3 consist of (i) functions: for the computation of patients warfarin daily/weekly maintenance dosage (MDD and MDW) and loading dosages for days 1, 2 and 3 (LC, LD1, LD2 and LD3), (ii) sub-protocols: secure multiplication

Table 3.3: Proposed protocols and functions used in our secure pharmacogenetic warfarin dosage prediction scheme

Definition	Notation	Functionality
Functions		
Daily maintenance dose function	MDD	Computation of MDD
Weekly maintenance dose function	MDW	Computation of MDW
Loading dose coefficient function	LC	Computation of LC for days 1, 2 and 3 of therapy
Loading dose day 1 function	LD1	Computation of LD1
Loading dose day 2 function	LD2	Computation of LD2
Loading dose day 3 function	LD3	Computation of LD3
Sub-protocols		
Secure multiplication protocol	SML	Multiplication of two encrypted numbers
Secure division protocol	SDV	Division of two encrypted numbers
Secure state comparison protocol	SSC	Compares states of patients data
Secure warfarin elimination rate protocol	SWR	Estimates warfarin elimination rate for days 1, 2 and 3
Main protocols		
Secure maintenance dose protocol	SMD	Estimates patients daily and weekly maintenance dose
Secure loading dose protocol	SLD	Estimates patients loading dose for days 1, 2 and 3

protocol (SML), secure division protocol (SDV), secure state comparison protocol (SSC) and secure warfarin elimination rate protocol (SWR), and (iii) main protocols: secure maintenance dosage protocol (SMD) and secure loading dosage protocol (SLD). Our sub-protocols are one round SMC-based protocols between the CP and the CSP, and their goal is to perform operations on ciphertext such as multiplication and comparison which are not feasible solely based on homomorphic computation. We first introduce and formulate these sub-protocols and then followed by the main protocols SMD and SLD. We present a workflow diagram to show how our secure protocols connect and invoke each other in Figure 3.4.

Secure Multiplication Protocol (SML)

The Paillier cryptosystem is additive homomorphic which means it does not support multiplication between ciphertext. To do this in our scheme, we propose **Algorithm**

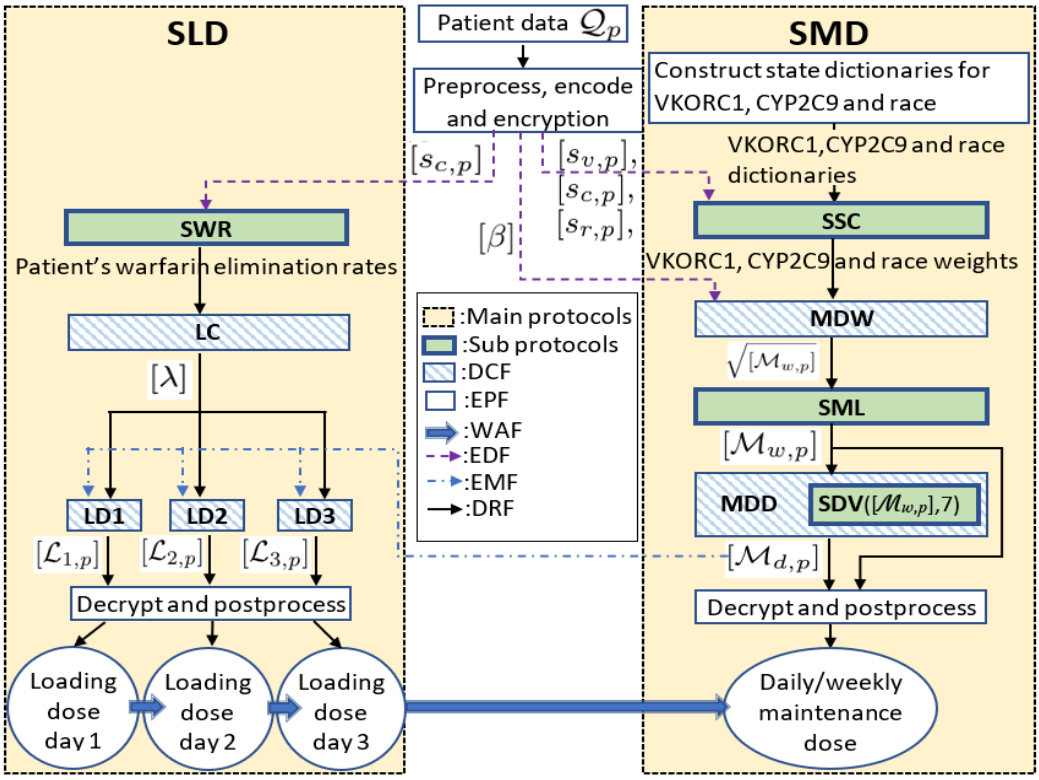


Figure 3.4: The workflow of protocols in our proposed secure pharmacogenetic warfarin dosage prediction scheme. ***Notations:*** **DRF**: Data/request flow; **DCF**: Secure dosage computation functions; **EPF**: Encryption/decryption, pre/post processing functions; **EDF**: Encrypted patients data flow; **EMF**: Encrypted patients daily maintenance dosage for loading dosage computation; **WAF**: Warfarin administration flow; $\boldsymbol{\mathcal{Q}_p}$: Patients data; $[\boldsymbol{s_{v,p}}]$: VKORC1 state; $[\boldsymbol{s_{c,p}}]$: CYP2C9 state; $[\boldsymbol{s_{r,p}}]$: Patients race state; $[\boldsymbol{\lambda}]$: coefficient of the estimated half-life of warfarin over the first three days; $[\boldsymbol{\beta}]$: Patients clinical data; $[\boldsymbol{\mathcal{L}_{1,p}}]$, $[\boldsymbol{\mathcal{L}_{2,p}}]$, $[\boldsymbol{\mathcal{L}_{3,p}}]$: Loading dosage days 1, 2 and 3; $[\boldsymbol{\mathcal{M}_{d,p}}]$: Daily maintenance dosage; $[\boldsymbol{\mathcal{M}_{w,p}}]$: Weekly maintenance dosage.

1. Given two encrypted genomic or clinical data values $[a]$ and $[b]$, the goal of SML is to compute $[a \times b]$ without compromising data privacy.

Secure Division Protocol (SDV)

Given an encryption of two genomic or clinical values $[a]$ and $[b]$, we compute an encryption of the division $[a]/[b]$ using the protocol from [129]. The division protocol [129] performs this task in a constant number of rounds while keeping the inputs private.

Secure State Comparison Protocol (SSC)

Secure integer comparison has always been a complex issue. State-of-the-art secure integer comparison solutions either incur high computation or communication cost due

Algorithm 1: Secure Multiplication Protocol (SML)

Input: CP: $[a], [b], sk_1$; CSP: sk_2

Output: CP: $[a \times b]$; CSP: $\perp$

1. **CP:** (a) Blindly creates two shares $[a_1], [a_2] \in \mathbb{Z}_N$ of $[a]$ and randomly chooses $r_1 \in \mathbb{Z}_N$, and computes: $a_1 \leftarrow r_1$, $[a_2] \leftarrow [a - r_1]$
   (b) Computes $X \leftarrow [b]^{a_1}$
   (c) Partially decrypts $[a_2]' \leftarrow PD_{sk_1}([a_2])$
   (d) Sends X and $[a_2]'$ to CSP
2. **CSP:** (a) Decrypts $[a_2]'$: $a_2 \leftarrow PD_{sk_1}([a_2]')$
   (b) Computes $Y \leftarrow [b]^{a_2}$
   (c) Sends Y to CP
3. **CP:** (a) Computes $[a \times b] \leftarrow X \times Y$
   $[a \times b] \leftarrow [b]^{a_1} \times [b]^{a_2} = [b]^{r_1} \times [b]^{(a-r_1)} = [r_1 b] \times [(a - r_1)b]$

to bit decomposition of integers and multiple rounds of communication [116, 113]. We propose an efficient SMC based comparison protocol using homomorphic and blinding techniques without bit decomposition. Given an encrypted SNP state $[s_i]$ for $i \in \{1, 2, \ldots, n\}$ of a patient, the CP requires an encryption of its corresponding weight $[w_i]$ to compute secure maintenance dosage. We do this with SSC where CP has access to the weights of all SNPs (i.e. variants of CYP2C9 and VKORC1) with their encoded states as shown in Table 3.2. It is important to note that SNP weights for a given pharmacogenomic model are public variables and for the case of IWPC model we refer the reader to [108] for the regression model details with SNP weights. With this in mind, CP has a dictionary object $Dic\{S, W\}$ of the SNP states $S = \{s_1, s_2, \ldots, s_n\}$ with their corresponding weights $W = \{w_1, w_2, \ldots, w_n\}$ and then encrypts it with public key pk. CP then securely computes the difference between $[S]$ and the SNP state $[s_i]$ to get updated dictionary $Dic\{[S_\varrho], [W]\}$ where $[S_\varrho] = [S - s_i]$. CP then blinds $[S_\varrho]$ and partially decrypts it before transmitting the updated dictionary $Dic\{[Z]', [W]\}$ to CSP. CSP then decrypts $[Z]'$ and gets the corresponding weight $[w]_\delta$ for which $z = 0$, and finally returns the refreshed cipher $[w]_\gamma$ to CP. Informally speaking, at the end of the protocol, the CP learns no sensitive patients data other than the encrypted weight $[w_i]$. Details of SSC protocol is proposed in **Algorithm 2**.

We remark that our sub-protocol SSC is efficient which will be discussed in Section 3.6 and involves only one round of communication, and comparing to state-of-the-art, [116] incurs 3 rounds of communications while [113] is not efficient.

Algorithm 2: Secure State Comparison Protocol (SSC)

Input: CP: $[s_i], Dic\{S, W\}, sk_1$; CSP: sk_2

Output: CP: $[w]_\gamma$; CSP: $\perp$

1. **CP:** (a) Encrypts Dic and computes the difference between $[S - s_i]$:

$$Dic\{[S_\varrho], [W]\} \leftarrow \begin{Bmatrix} [s_1 - s_i], & [w_1] \\ [s_2 - s_i], & [w_2] \\ \vdots & \vdots \\ [s_q - s_i], & [w_q] \end{Bmatrix}$$

(b) Randomly chooses $r_1, r_2, \ldots, r_q \in \mathbb{Z}_N$, blinds the states $[S_\varrho]$ to get blinded dictionary $Dic\{[Z], [W]\}$, and partially decrypts them with sk_1:

$$Dic\{[Z]', [W]\} \leftarrow \begin{Bmatrix} [z_1]' \leftarrow PD_{sk_1}([z_1], & [w_1] \\ [z_2]' \leftarrow PD_{sk_1}([z_2], & [w_2] \\ \vdots & \vdots \\ [z_q]' \leftarrow PD_{sk_1}([z_3], & [w_q] \end{Bmatrix}$$

(c) Sends $Dic\{[Z]', [W]\}$ to CSP

2. **CSP:** (a) Decrypts $[Z]'$ in $Dic\{[Z]', [W]\}$ with sk_2
$Dic\{Z, [W]\} \leftarrow Dic\{PD_{sk_2}([Z]'), [W]\}$
(b) Get weight $[w]_\delta$ from $Dic\{Z, [W]\}$ for which $z = 0$

```
for i = 0 to p do
    if z_i = 0 then
        [w]_δ ← [w_i]
        exit for
    end if
end for
```

(c) Re-encrypt $[w]_\delta$: $[w]_\gamma \leftarrow CR([w]_\delta)$
(d) Sends $[w]_\gamma$ to CP

3.4.3 Secure Maintenance Dose Protocol (SMD)

Patients maintenance dosage for warfarin is estimated based on the regression model presented in [108, 114, 115]. We reformulate **BM** (i.e. Equation 3.2) to propose our secure maintenance dosage computation Equation 3.4 (**MDW**) to securely compute the weekly maintenance dosage for patient p without revealing sensitive data.

$$[\mathcal{M}_{w,p}] = \left([c'] + \sum_{i=1}^{n} [\beta]_{i,p}^{u'_i} + \sum_{j=1}^{m} [w']_j [S]_{i,p} \right)^2 \qquad (3.4)$$
$$i \in \{1, 2 \ldots n\}; j \in \{1, 2 \ldots m\}$$

where c is a constant, β is p's clinical variables {age, height, weight, race, enzyme inducer status, amiodarone status}, u is the weights of β, S is p's SNP states and w is the weights of S. Since c, u and w are real numbers and cannot be computed by Paillier and most cryptosystems as discussed in Subsection 3.4.1, we round them off to integers using the scaling technique given in Subsection 3.4.1. This gives us the scaled versions i.e. $c' = c \times \Psi_d$, $u' = u \times \Psi_d$ and $w' = w \times \Psi_d$ (where $\Psi_d = 10^d$) which

we use in our secure protocol. The rounding-off, however, results in the rounding-off error $\mathcal{E}_r$ and data losses or accuracy losses $\mathcal{A}_l$.

Theorem 1 *The round-off error $\mathcal{E}_r$ and accuracy losses $\mathcal{A}_l$ in our proposed protocol **SMD** which is as a result of converting real-valued weights c, u and w to integers c', u' and w' respectively by the scale factor Ψ_d is negligible.*

Proof 1 *We prove that the accuracy losses $\mathcal{A}_l$ incurred by our protocol **SMD** is negligible and a function of $\mathcal{E}_r$.*
For the sake of brevity, since c', u' and w' are factors of $\Psi_d = 10^d$, we replace $\left([c'] + \sum_{i=1}^{n}[\beta]_{i,p}^{u'_i} + \sum_{j=1}^{m}[w']_j[S]_{i,p}\right)^2$ from Equation 3.4 with $(\Delta + \mathcal{E}_r) \times 10^d$. This gives: $\mathcal{M}' = \left(\left(c + \sum_{i=1}^{n} u_i\beta_{i,p} + \sum_{j=1}^{m} w_j S_{i,p} + \mathcal{E}_r\right) \times 10^d\right)^2$; where $\mathcal{M}'$ is our estimated maintenance dosage. Simplifying this Equation by replacing $(c + \sum_{i=1}^{n} u_i\beta_{i,p} + \sum_{j=1}^{m} w_j S_{i,p})$ with $\mathcal{M}$ (i.e. maintenance dosage ground truth) and ignoring the squared gives: $\mathcal{M}' = (\mathcal{M} + \mathcal{E}_r) \times 10^d$; now descaling by 10^d (i.e. dividing by 10^d) and solving for $\mathcal{E}_r$ as a function of $\mathcal{M}$ and $\mathcal{M}'$ gives:

$$\mathcal{E}_r = (\mathcal{M} - \mathcal{M}') \tag{3.5}$$

where $|(\mathcal{M} - \mathcal{M}')|$ is accuracy loss $\mathcal{A}_l$ between our estimated maintenance dosage $\mathcal{M}'$ and ground truth maintenance dosage $\mathcal{M}$. We know from Equation 3.5 that $\mathcal{A}_l = \mathcal{E}_r$. Now substituting $\mathcal{E}_r$ from Equation 3.3 gives: $\mathcal{A}_l = |1/2 \times 10^{-d}|$ Since $\mathcal{A}_l$ will always be negligible for increasing d, we can conclude that $\mathcal{A}_l$ is also bounded by $|1/2 \times 10^{-d}|$ i.e. $\mathcal{A}_l \leq |1/2 \times 10^{-d}|$

Patient p's daily maintenance dosage $\mathcal{M}_{d,p}$ is then computed with Equation 3.6 (**MDD**) as follows.

$$[\mathcal{M}_{d,p}] = [\mathcal{M}_{w,p}]/7 \tag{3.6}$$

Now using Equations 3.4 and 3.6, and the sub-protocols - **SSC** and **SML**, we formulate our secure maintenance dosage protocol (**SMD**) as follows.

Step 1: The MU wishes to know the maintenance dosage of p and sends a request to the CP for computation.

Step 2: CP collaborates with CSP to execute SMD with inputs @CP: patient p's encrypted data $[\mathcal{Q}_p], sk_1$ and @CSP: sk_2, and returns $[\mathcal{M}_{d,p}]$ and $[\mathcal{M}_{w,p}]$ to MU.

Step 3: MU then decrypts $[\mathcal{M}_{d,p}]$ and $[\mathcal{M}_{w,p}]$ and post-processes them with Ψ_d to obtain the daily and weekly maintenance dosage respectively. Details are shown in **Algorithm 3**.

Algorithm 3: Secure Maintenance Dose Protocol (SMD)

Input: CP: $[\mathcal{Q}_p], sk_1$; CSP: sk_2

Output: CP: $[\mathcal{M}_{d,p}]$, $[\mathcal{M}_{w,p}]$; CSP: $\perp$

1. **CP:** (a) Constructs and populates $Dic\{S_v, W_v\}, Dic\{S_c, W_c\}$ and $Dic\{S_r, W_r\}$; where S_v, S_c and S_r are the states of VKORC1, CYP2C9 and Race, and their respective weights are W_v, W_c and W_r.
   (b) CP then initiates SSC protocol with CSP.
2. **CP and CSP:** (a) Execute $[w_v]_\gamma \leftarrow \mathbf{SSC}([s_{v,p}], Dic\{S_v, W_v\}$.
   (b) Execute $[w_c]_\gamma \leftarrow \mathbf{SSC}([s_{c,p}], Dic\{S_c, W_c\}$.
   (c) Execute $[w_r]_\gamma \leftarrow \mathbf{SSC}([s_{r,p}], Dic\{S_r, W_r\}$.
3. **CP:** (a) Calculate patient p's weekly maintenance dosage.
   $[\Upsilon] \leftarrow \mathbf{MDW}(P, [w_v]_\gamma, [w_c]_\gamma, [w_r]_\gamma)$
   $[\mathcal{M}_{w,p}] \leftarrow \mathbf{SML}([\Upsilon], [\Upsilon])$
   (b) Compute p's daily maintenance dosage $[\mathcal{M}_{d,p}] \leftarrow \mathbf{MDD}([\mathcal{M}_{w,p}])$.
   In the process **MDD** invokes **SDV** for secure division operation.
   (c) Return $[\mathcal{M}_{d,p}]$, $[\mathcal{M}_{w,p}]$ to MU.
4. **MU:** (a) Decrypt $[\mathcal{M}_{d,p}]$ and $[\mathcal{M}_{w,p}]$ using $\mathcal{M}_{d,p} \leftarrow \mathrm{D}([\mathcal{M}_{d,p}])$ and $\mathcal{M}_{w,p} \leftarrow \mathrm{D}([\mathcal{M}_{w,p}])$ respectively.
   (b) Postprocess $\mathcal{M}_{d,p}$ and $\mathcal{M}_{w,p}$ with scale factor Ψ_d to obtain patient p's daily and weekly maintenance dosage as $\mathcal{M}'_{d,p} \leftarrow \mathcal{M}_{d,p}/\Psi_d$ and $\mathcal{M}'_{w,p} \leftarrow \mathcal{M}_{w,p}/\Psi_d$ respectively.

3.4.4 Secure Loading Dose Protocol (SLD)

As discussed in Section. 3.2.1, the loading dosage is derived from the maintenance dosage as a function of the estimated half-life of warfarin. For secure domain computation, we propose Equation 3.7 which is also a function of the estimated half-life and an adaptation of the loading dosage pharmacogenetic model **LMD** [114] (i.e. Equation 3.1). However different from [114], we first compute the warfarin elimination rates $\vartheta_1 = \mathrm{e}^{kt}$, $\vartheta_2 = \mathrm{e}^{2kt}$, $\vartheta_3 = \vartheta_1\vartheta_2$ for days 1, 2 and 3 respectively (k is the warfarin elimination rate constant which is dependent on the CYP2C9 genotype and t is the elimination time interval in hours which is a day for warfarin i.e. 24hrs). We then evaluate λ (a value we call the coefficient of the estimated half-life of warfarin over the first three days of starting therapy, as depicted in Figure. 3.5) and we finally compute the loading dosages by expressing the daily MD as a fraction of these coefficients. Equation 3.7 is used to compute $[\lambda]$.

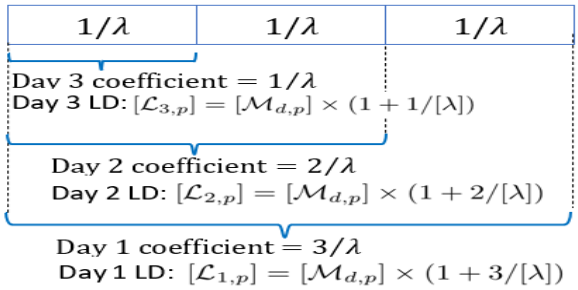


Figure 3.5: Loading dosage (LD) coefficients for days 1, 2 and 3 used in our secure loading dosage computation protocol. λ is the coefficient of the estimated half-life of warfarin over the first three days of initiating therapy.

$$[\lambda] = \left(([\vartheta'_1] + [\vartheta'_2] + [\vartheta'_3] - (3 \times 10^d)\right)^{-1} \tag{3.7}$$

where λ is the loading dosage coefficient over the first three days. We represent Equation 3.7 by the function $\mathbf{LC}([\vartheta'_1], [\vartheta'_2], [\vartheta'_3])$. We propose the sub-protocol secure warfarin elimination rate computation (**SWR**), which is an extension of **SSC**, to compute the elimination rates $[\vartheta'_1], [\vartheta'_2], [\vartheta'_3]$. Details of **SWR** is proposed in **Algorithm 4**. It is important to note that ϑ_1, ϑ_2 and ϑ_3 are real numbers, and we round them off to integers with the scale factor Ψ_l i.e. $\vartheta'_1 = \vartheta_1 \times \Psi_l$, $\vartheta'_2 = \vartheta_2 \times \Psi_l$ and $\vartheta'_3 = \vartheta_1 \times \vartheta_2 \times \Psi_l$; where $\Psi_l = 10^d$. As discussed in the previous section, rounding-off may incur the rounding-off error $\mathcal{E}_r$ and data losses.

Corollary 1 *Our proposed scheme for secure loading dosage computation (**SLD**) incurs negligible round-off error $\mathcal{E}_r$ and data losses as a result of scaling the real-valued warfarin elimination rates ϑ_1, ϑ_2 and ϑ_3 with the scale factor Ψ_l to integers $\vartheta'_1, \vartheta'_2$ and ϑ'_3.*

Proof 2 *The proof follows the same lines as Theorem 1.*

Deriving our novel secure equations for loading dosages day 1, 2 and 3 based on λ and $\mathcal{M}_{d,p}$ gives the following: $[\mathcal{L}_{1,p}] = [\mathcal{M}_{d,p}] \times (1 + 3/[\lambda])$; $[\mathcal{L}_{2,p}] = [\mathcal{M}_{d,p}] \times (1 + 2/[\lambda])$; and $[\mathcal{L}_{3,p}] = [\mathcal{M}_{d,p}] \times (1 + 1/[\lambda])$ which we represent with the functions $\mathbf{LD1}([\mathcal{M}_{d,p}], [\lambda])$, $\mathbf{LD2}([\mathcal{M}_{d,p}], [\lambda])$ and $\mathbf{LD3}([\mathcal{M}_{d,p}], [\lambda])$ respectively. We now formulate our secure loading dosage computation protocol (SLD) based on the functions **LC**, **LD1**, **LD2** and **LD3** and sub-protocol **SWR** as follows.

Step 1: The MU sends a request to the CP to compute p's loading dosages for days 1, 2 and 3.

Algorithm 4: Secure Warfarin Elimination Rate Protocol (SWR)

Input: CP: $[s_{c,p}], Dic\{S_c, \vartheta_c\}, sk_1$; CSP: sk_2

Output: CP: $\langle[\vartheta'_1], [\vartheta'_2], [\vartheta'_3]\rangle_p$; CSP: $\perp$

1. **CP:** (a) Prepares $Dic\{S_c, \mathcal{V}_c\}$

   $S_c = \{s_1, s_2, \ldots, s_n\} = s_i$ for $i \in \{1, 2, \ldots, n\}$ i.e. a set of the states of CYP2C9 genotype from Table. 3.2, and $\mathcal{V}_c = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n\}$ is a set a tuples of elimination rates corresponding to S_c. That is $\mathcal{V}_i = \langle\vartheta_{1i}, \vartheta_{2i}, \vartheta_{3i}\rangle$ for $i \in \{1, 2, \ldots, n\}$, where $\vartheta_{1i}, \vartheta_{2i}$ and ϑ_{3i} are warfarin elimination rates for the ith SNP (i.e. s_i) of CYP2C9 for day 1, 2 and 3 respectively. $\vartheta_{1i} = \mathrm{e}^{k_i t}$, $\vartheta_{2i} = \mathrm{e}^{2k_i t}$, $\vartheta_{3i} = \vartheta_{1i}\vartheta_{2i}$, where k_i is warfarin elimination constant for s_i and $t = 24$hrs.

$$Dic\{S_c, \mathcal{V}_c\} \leftarrow \begin{Bmatrix} s_1, & \langle\vartheta_{11}, \vartheta_{21}, \vartheta_{31}\rangle \\ s_2, & \langle\vartheta_{12}, \vartheta_{22}, \vartheta_{32}\rangle \\ \vdots & \vdots \\ s_n, & \langle\vartheta_{1n}, \vartheta_{2n}, \vartheta_{3n}\rangle \end{Bmatrix}$$

   (b) Scale $\mathcal{V}_c$ and encrypt $Dic\{S_c, \mathcal{V}'_c\}$.

   Scaling $\mathcal{V}_c$ with factor $\Psi_l = 10^d$ i.e. $\mathcal{V}'_c = \{\mathcal{V}'_1, \mathcal{V}'_2, \ldots, \mathcal{V}'_n\}$; $\mathcal{V}'_i = \langle\vartheta'_{1i}, \vartheta'_{2i}, \vartheta'_{3i}\rangle$, where $\vartheta'_{1i} = \vartheta_{1i} \times \Psi_l$, $\vartheta'_{2i} = \vartheta_{2i} \times \Psi_l$ and $\vartheta'_{3i} = \vartheta_{1i} \times \vartheta_{2i} \times \Psi_l$.

   $\mathcal{V}'_c \leftarrow \mathcal{V}_c \times \Psi_l$. Encrypt $Dic\{S_c, \mathcal{V}'_c\}$ with sk_1 to give below

$$Dic\{[S_c], [\mathcal{V}'_c]\} \leftarrow \begin{Bmatrix} [s_1], & \langle[\vartheta'_{11}], [\vartheta'_{21}], [\vartheta'_{31}]\rangle \\ [s_2], & \langle[\vartheta'_{12}], [\vartheta'_{22}], [\vartheta'_{32}]\rangle \\ \vdots & \vdots \\ [s_n], & \langle[\vartheta'_{1n}], [\vartheta'_{2n}], [\vartheta'_{3n}]\rangle \end{Bmatrix}$$

   (c) CP then initiates **SSC** protocol with CSP.

2. **CP and CSP:**

   (a) Execute $\langle[\vartheta'_1], [\vartheta'_2], [\vartheta'_3]\rangle_p \leftarrow \mathbf{SSC}([s_{c,p}], Dic\{[S_c], [\mathcal{V}'_c]\})$

   (b) Finally, CSP returns Patient p's warfarin elimination rates $\langle[\vartheta'_1], [\vartheta'_2], [\vartheta'_3]\rangle_p$ for $[s_{c,p}]$ to CP.

Step 2: CP collaborates with CSP to execute SLD and returns $[\mathcal{L}_{1,p}]$, $[\mathcal{L}_{2,p}]$ and $[\mathcal{L}_{3,p}]$ to MU.

Step 3: MU then decrypts $[\mathcal{L}_{1,p}]$, $[\mathcal{L}_{2,p}]$ and $[\mathcal{L}_{3,p}]$ and postprocesses the result with Ψ_l to obtain p's loading dosages for days 1,2 and 3. The details are presented in **Algorithm 5**.

3.5 Security Analysis

In this section, we analyze the security of our sub-protocols (**SML**, **SSC** and **SWR**) and main protocols (**SMD** and **SLD**). Finally, based on the security of these protocols, we evaluate the security of our secure framework as a whole.

Algorithm 5: Secure Loading Dose Protocol (SLD)

Input: CP: $[\mathcal{Q}_p], sk_1, [\mathcal{M}_{d,p}]$; CSP: sk_2

Output: CP: $[\mathcal{L}_{1,p}], [\mathcal{L}_{2,p}], [\mathcal{L}_{3,p}]$; CSP: $\perp$

1. **CP:** (a) Securely obtains patient p's warfarin elimination rates $[\vartheta'_1], [\vartheta'_2]$ and $[\vartheta'_3]$ for $[s_{c,p}]$.
   $\langle [\vartheta'_1], [\vartheta'_2], [\vartheta'_3] \rangle \leftarrow \mathbf{SWR}([s_{c,p}])$
   (b) Compute the coefficient of the estimated half-life of warfarin over the first three days as: $[\lambda] \leftarrow \mathbf{LC}([\vartheta'_1], [\vartheta'_2], [\vartheta'_3])$
   (c) Compute p's loading dosages for days 1, 2 and 3.
   LD1, **LD2** and **LD3** invokes **SDV** for secure division operation.
   $[\mathcal{L}_{1,p}] \leftarrow \mathbf{LD1}([\mathcal{M}_{d,p}], [\lambda])$
   $[\mathcal{L}_{2,p}] \leftarrow \mathbf{LD2}([\mathcal{M}_{d,p}], [\lambda])$
   $[\mathcal{L}_{3,p}] \leftarrow \mathbf{LD3}([\mathcal{M}_{d,p}], [\lambda])$
   (d) Return $[\mathcal{L}_{1,p}], [\mathcal{L}_{2,p}], [\mathcal{L}_{3,p}]$ to MU.
4. **MU:** (a) Decrypt $[\mathcal{L}_{1,p}], [\mathcal{L}_{2,p}]$ and $[\mathcal{L}_{3,p}]$ using $\mathcal{L}_{1,p} \leftarrow \mathrm{D}([\mathcal{L}_{1,p}])$, $\mathcal{L}_{2,p} \leftarrow \mathrm{D}([\mathcal{L}_{2,p}])$, and $\mathcal{L}_{3,p} \leftarrow \mathrm{D}([\mathcal{L}_{3,p}])$ respectively
   (b) Postprocess $\mathcal{L}_{1,p}$, $\mathcal{L}_{2,p}$ and $\mathcal{L}_{3,p}$ with scale factor Ψ_l to obtain p's actual loading dosages as $\mathcal{L}'_{1,p} \leftarrow \mathcal{L}_{1,p}/\Psi_l$, $\mathcal{L}'_{2,p} \leftarrow \mathcal{L}_{2,p}/\Psi_l$, and $\mathcal{L}'_{3,p} \leftarrow \mathcal{L}_{3,p}/\Psi_l$ respectively

3.5.1 Security of Sub-protocols

We start the analysis of our protocols with a formal definition for a secure protocol under a semi-honest model [15].

Definition 1 (security in the semi-honest model [15]) *Let $\mathcal{F}(.)$ be a deterministic function and π be a protocol between n parties $p_1, \ldots, p_n$. Let $VIEW^{\pi}_{p_i}$ represent the view of party p_i during the execution of protocol π. We say that π securely computes $\mathcal{F}(.)$ under the semi-honest model if for each party p_i there exists a polynomial-time algorithm $SIM^{\pi}_{p_i}$ that simulates its view such that $SIM^{\pi}_{p_i}(I_{p_i}, \mathcal{F}(.)) \overset{c}{\approx} VIEW^{\pi}_{p_i}(I_{p_i}, O_{p_i})$; where $\overset{c}{\approx}$ denotes computationally indistinguishable, I_{p_i} is a set of p_i's private input and O_{p_i} is set of outputs for p_i from a collaborating party.*

Theorem 2 *The **SML** protocol described in Section 3.4.2 securely computes multiplication over ciphertext in the presence of semi-honest (non-colluding) adversaries.*

Proof 3 *We provide proof to show that the simulators of CP (SIM^{SML}_{CP}) and CSP (SIM^{SML}_{CSP}) are computationally indistinguishable from their views $VIEW^{SML}_{CP}$ and $VIEW^{SML}_{CSP}$ respectively. The view of CP from Algorithm 1 consists of its private inputs and the messages it receives from CSP i.e. $VIEW^{SML}_{CP} = \{([a], [b]), Y\}$. Now constructing a simulator for $VIEW^{SML}_{CP}$ will give $SIM^{SML}_{CP} = \{([\hat{a}], [\hat{b}]), \hat{Y}\}$; where $\hat{Y} = [\hat{b}]^{\hat{a}_2}$ and*

$\hat{a}, \hat{b}, \hat{a}_2$ are randomly generated from $\mathbb{Z}_n$. Since elements of $VIEW_{CP}^{SML}$ and SIM_{CP}^{SML} are encrypted with the Paillier cryptosystem which is semantically secure, we can conclude that $VIEW_{CP}^{SML}$ is computationally indistinguishable from SIM_{CP}^{SML}. Likewise, the view of CSP is $VIEW_{CSP}^{SML} = \{[a_2]', [b], X\}$ and its simulated image will produce $SIM_{CSP}^{SML} = \{[\hat{a}_2]', [\hat{b}], \hat{X}\}$; where $\hat{X} = [\hat{b}]^{\hat{a}_1}$, $[\hat{a}_2]' = PD_{sk_1}([\hat{a}_2])$, $\hat{a}_1$ and $\hat{a}_2$ are random shares of $\hat{a}$ (i.e. $\hat{a}_1 + \hat{a}_2 = \hat{a}$), and $\hat{a}, \hat{b}$ are randomly generated from $\mathbb{Z}_n$. Again, due to the semantic security of the Paillier cryptosystem, $[a_2]', [b]$ and X are indistinguishable from $[\hat{a}_2]', [\hat{b}]$ and $\hat{X}$ respectively, and consequently $VIEW_{CSP}^{SML}$ is computationally indistinguishable from SIM_{CSP}^{SML}.

The security proofs of **SSC** and **SWR** follow the same lines as **SML** under the semi-honest model where, for each protocol, there exists a polynomial-time algorithm to simulate its view. However, for the security proof of **SDV**, we refer the reader to [129] which is also semantically secure.

3.5.2 Security of Main Protocols

Here we provide security proofs for our main protocols **SMD** and **SLD** to conform to Definition 1 under the semi-honest model.

Theorem 3 *The **SMD** protocol described in Section 3.4.3 securely computes patients maintenance dosages (daily and weekly) under the non-colluding semi-honest adversarial model.*

Proof 4 *The **SMD** protocol consists of (i) sub-protocols **SML**, **SDV** and **SSC** which are secure as already proven earlier, and (ii) local data processing in encrypted form on the CP which are non-interactive. That is to say no patients genomic and clinical data is leaked to CP or any external party. Based on this we conclude that our **SMD** protocol is secure.*

The security proof of **SLD** protocol is similar to that of the **SMD** protocol under the semi-honest model.

3.5.3 Security of the Entire Framework

Here we analyze and discuss the security of our proposed framework for warfarin dosage prediction with respect to our secure computation protocols in the presence of an adversary who is likely to eavesdrop on the communication channels between parties in the framework. The security requirement for data confidentiality is met by our proposed scheme as (i) sensitive patients data are secured with Paillier encryption which is semantically secure and computation is performed using addition homomorphic properties, and (ii) our SMC-based protocols between CP and CSP do not reveal confidential data to unauthorized parties as already established from Theorems 2 and 3. Output privacy is achieved by our scheme due to the fact that the computed patients dosages is transmitted to the recipient in encrypted form and as such, it is not leaked to any unintended party.

3.6 Experimental Analysis

In this section, we discuss the experiment setup of our scheme and analyze its performance. We implement our framework with C# and run the experiments on a PC with 2.60 GHz processor and 16 GB memory. For the modified Paillier encryption, we denote n as 1024 bits to achieve a security level of 80 bits. We run our experiments on a real patients dataset [130] to predict their warfarin maintenance and loading dosages for days 1, 2 and 3. Our dataset consist of 4,043 geographically diverse patients (Whites, Asians, Caucasians, and Africans) from four continents (Europe, North America, Asia and South America). Each patient has clinical data (age, height, weight, race, enzyme inducer status and amiodarone status) and genomic data (CYP2C9 and VKORC1). We then evaluate the performance of our scheme based on its computation and communication cost and analyze the accuracy of the predicted dosages by comparing them to the baseline dosages (i.e. patients dosages computed without security).

Table 3.4: Runtime in milliseconds (ms) to securely compute warfarin dosage

Procedure	**Parties**				**Total**
	MU	**SU**	**CP**	**CSP**	
P&E	82.33	27.14	NA	NA	109.47
SML	NA	NA	69.83	53.64	123.47
SWR	NA	NA	393.98	80.13	474.11
SSC_V	NA	NA	224.52	59.65	284.17
SSC_C	NA	NA	397.8	54.29	452.09
SSC_R	NA	NA	223.96	38.93	262.89
SMD	53.03	NA	1149.27	210.48	1412.78
SLD	79.96	NA	1035	190.85	1305.81
D&P	132.99	NA	NA	NA	132.99

Notations: **P&E**: Preprocessing and encryption; **D&P**: Decryption and postprocessing; **NA**: Not applicable where the procedure does not run on a specified party.

3.6.1 Computation Analysis

Here we analyze and discuss the runtime and computation complexity of our proposed protocols for warfarin dosage prediction. Table 3.4 shows the average runtime (over 10 runs) for our protocols on each party (MU, SU, CP and CSP) within our framework. The table suggests that the runtime on the cloud servers (CP and CSP) for our proposed protocols (**SML**, **SWR**, **SSC**, **SMD** and **SLD**) is higher than that of the client side (MU and SU). Furthermore, it is important to note that the preprocessing and encryption operations (P&E) on the MU (for patients clinical data) and the SU (for patients genomic data) which runs in $82.33ms$ and $27.14ms$ respectively are one-time operations, meaning that patients data can be encrypted once and processed many times in the cloud. Since they are one-time operations, they can be performed offline. This leaves our scheme with a client-side operation of only $132.99ms$ during the decryption and postprocessing (D&P) operation at the MU, i.e. $\tau_s > \tau_c$; where $\tau_s = 2585.6ms$ is the total runtime cost on the cloud servers (CP and CSP) and $\tau_c = 132.99ms$ is the total runtime cost on the client side (MU and SU). Informally, our observation of a higher runtime cost on the cloud servers (CP and CSP) than the clients (MU and SI) is as expected in an outsourced computation model [125] where the goal of a resource constraint client such as a medical facility is to offload computation to a high-end computation environment such as the cloud.

Table 3.5: Comparison between our secure and baseline computed dosages

Pat	MD weekly			MD daily			LD day 1			LD day 2			LD day 3		
	MDW (mg)	**BM_w (mg)**	**%Err**	**MDD (mg)**	**BM_d (mg)**	**%Err**	**LD1 (mg)**	**BL_1 (mg)**	**%Err**	**LD2 (mg)**	**BL_2 (mg)**	**%Err**	**LD3 (mg)**	**BL_3 (mg)**	**%Err**
1	40.0968	40.0966	5.19E-04	5.7281	5.7281	1.42E-04	9.1992	9.1988	5.14E-03	8.0422	8.0417	6.33E-03	6.8852	6.8846	7.92E-03
2	35.7975	35.7949	7.36E-03	5.1139	5.1139	5.19E-04	8.2129	8.2124	5.52E-03	7.1799	7.1794	6.71E-03	6.1469	6.1464	8.29E-03
3	48.4764	48.4768	7.82E-04	6.9252	6.9252	1.29E-05	11.1217	11.1212	5.01E-03	9.7229	9.7223	6.20E-03	8.3241	8.3234	7.79E-03
4	35.0049	35.0023	7.70E-03	5.0007	5.0007	2.06E-04	8.03104	8.0306	5.21E-03	7.0209	7.0205	6.39E-03	6.0108	6.0103	7.98E-03
5	23.912	23.9125	1.67E-03	3.4160	3.416	4.18E-04	7.0196	7.0195	7.28E-04	5.8184	5.8185	1.29E-03	4.6172	4.6171	3.04E-03
6	55.3328	55.3325	5.66E-04	7.9047	7.9046	1.03E-03	12.6948	12.694	6.03E-03	11.0981	11.0973	7.22E-03	9.5014	9.5005	8.80E-03
7	33.6759	33.6746	4.14E-03	4.8109	4.8108	1.10E-03	8.6955	8.6955	1.86E-04	7.4006	7.4005	2.55E-03	6.1058	6.1054	5.92E-03
8	43.6485	43.6479	1.47E-03	6.2355	6.2354	1.57E-03	10.0141	10.0134	6.57E-03	8.7546	8.7539	7.76E-03	7.4950	7.49433	9.34E-03
9	32.9017	32.9048	9.31E-03	4.7002	4.7002	9.00E-04	9.7683	9.7684	1.68E-03	8.0789	8.0787	2.70E-03	6.3896	6.3895	2.04E-03
10	24.8442	24.8459	6.45E-03	3.5492	3.5491	2.19E-03	7.2932	7.2931	2.50E-03	6.0452	6.0452	4.79E-04	4.7972	4.7969	4.81E-03
	Avg %Acc: 99.9953%			**Avg %Acc:** 99.9992%			**Avg %Acc:** 99.9961%			**Avg %Acc:** 99.9952%			**Avg %Acc:** 99.9934%		

Notations: **Acc**: Accuracy; **Avg**: Average; **BM_d**: Baseline daily maintenance dosage; **BM_w**: Baseline weekly maintenance dosage; **BL_1**: Baseline loading dosage day 1; **BL_2**: Baseline loading dosage day 2; **BL_3**: Baseline loading dosage day 3; **Err**: Error; **LD**: Loading dosage; **mg**: milligrams; **MD**: Maintenance dosage; **MSE**: Mean square error; **Pat**: Patient; **MDW**: Secure weekly maintenance dosage; **MDD**: Secure daily maintenance dosage; **LD1**: Secure loading dosage day 1; **LD2**: Secure loading dosage day 2; **LD3**: Secure loading dosage day 3

3.6.2 Communication Overhead

In this section, we discuss the communication overhead incurred by our proposed secure protocols. In Table 3.6, we present the communication overhead as data transmitted in bits between two parties at a time i.e. i) between MU and CP (MU-CP), ii) CP and CSP (CP-CSP), and iii) SU and CP (SU-CP). Data transmitted between parties in our framework is in encrypted form (ciphertext pair) which is bounded by n from the encryption in Section 3.2.2 i.e. encrypted patients data is within the domain $\mathbb{Z}_n$, consequently the number of bits ρ required to transmit a ciphertext pair is given as $\rho \leq 2log_2 n$. Table 3.6 shows the data transmitted as a function of ρ (i.e. number of ciphertext transmitted between parties $\times\rho$). For example, for the protocol P&E data transmitted between (i) MU and CP (MU-CP) is the number of encrypted patients clinical data sent from MU to CP i.e. $\varphi_d\rho$; where φ_d is the number of patients clinical data variables such as age, height, etc., and (ii) SU and CP (SU-CP) is the number of encrypted patients SNPs sent from SU to CP i.e. $\varphi_s\rho$; where φ_s is the number of SNPs. It is evident from Table 3.6 that the communication cost between CP and CSP is higher than that between MU and CP, and SI and CP. This is a result of the number of ciphertext transmitted between CP and CSP within one round of SMC computation. We point out that our proposed SMC-based protocols are one round communication protocols. The highest of these costs is the total bits transmitted for SLD which is $6\varphi_c\rho + 13\rho = 100,352$ bits (for n with bit length of 1024 bits, $\rho = 2048$ bits and $\varphi_c = 6$). Now converting 100,352 bits to KB gives 12.544 KB data transferred which is feasible for health and clinical applications [131].

3.6.3 Accuracy Analysis

In pharmacogenomics, accuracy is crucial when it comes to prescribing a drug based on an individual's genome as the wrong dosage can be fatal. As a result, we need to evaluate the accuracy of the computed warfarin maintenance and loading dosages from our proposed scheme compared to baseline computation from the baseline models **BL** and **BM** [114, 115] (i.e. computing the dosages without security applied – plaintext domain computation). In this section, we discuss our accuracy analysis and present

Table 3.6: Communication overhead (bits) incurred by our proposed secure protocols to compute warfarin dosage

Protocol	**Parties**			**Total**
	MU-CP	**CP-CSP**	**SU-CP**	
P&E	$\varphi_d\rho$	NA	$\varphi_s\rho$	$\varphi_d\rho + \varphi_s\rho$
SML	NA	4ρ	NA	4ρ
SWR	NA	$6\varphi_c\rho + 6\rho$	NA	$6\varphi_c\rho + 6\rho$
SSC_V	NA	$2\varphi_v\rho + \rho$	NA	$2\varphi_v\rho + \rho$
SSC_C	NA	$2\varphi_c\rho + \rho$	NA	$2\varphi_c\rho + \rho$
SSC_R	NA	8ρ	NA	8ρ
SMD	ρ	$2\varphi_v\rho+2\varphi_c\rho+14\rho$	NA	$2\varphi_v\rho+2\varphi_c\rho+15\rho$
SLD	3ρ	$6\varphi_c\rho + 10\rho$	NA	$6\varphi_c\rho + 13\rho$
D&P	4ρ	NA	NA	4ρ

Notations: **P&E**: Preprocessing and encryption; **D&P**: Decryption and postprocessing; $\boldsymbol{\varphi_d}$: number of patients clinical data variables; $\boldsymbol{\varphi_s}$: number of SNPs; $\boldsymbol{\varphi_v}$: number of VKORC1-rs9923231 polymorphisms; $\boldsymbol{\varphi_c}$: number of CYP2C9 polymorphisms; **NA**: Not applicable where the procedure does not run on a specified party.

the results in Table 3.5. We start by computing the Euclidean distance between the computed dosages from our scheme and that from baseline computation. The equation we use to compute Euclidean distance is given as: $\Gamma(D_p, \hat{D}_p) = \sqrt{(D_p - \hat{D}_p)^2}$; where $\Gamma(.)$ is the Euclidean distance function, D_p is p's dosage computed from our scheme and $\hat{D}_p$ is the baseline computed dosage for p. We then compute the percentage error (%Err) between our scheme and the baseline dosage as: $\%\text{Err} = \dfrac{\Gamma(D_p, \hat{D}_p)}{D_p} \times 100$. Finally, the percentage accuracy (%Acc) is computed as $\%\text{Acc} = 100 - \%\text{Err}$. The percentage error and accuracy from our results in Table 3.5 suggest that the dosages computed from our proposed secure scheme are identical to those from the baseline computation with negligible accuracy losses ($< 0.0066\%$). These losses translate to a difference of less than 10^{-4}mg between our proposed scheme and baseline computation which is a result of rounding real-valued patients data to integer as discussed in Section 3.4.1. It is important to note that in real life, prescribed warfarin tablets are rounded to the nearest 0.5mg [132]. This means for rounded dosages to the nearest 0.5mg, our scheme will produce the same results as the baseline dosages without any losses. For example, let's look at patient 1 from Table 3.5, rounding LD day 1 to the nearest 0.5mg.

1. Dose from our proposed scheme: $\lfloor 9.19923 \rfloor_{0.5\text{mg}} = 9\text{mg}$

2. Dose from baseline computation: $\lfloor 9.19876 \rfloor_{0.5\text{mg}} = 9\text{mg}$

where $\lfloor . \rfloor_{0.5\text{mg}}$ is a function to round the prescribed warfarin dosages to the nearest 0.5mg. This observation supports the fact that clinical decisions which are based on the warfarin prescribed dosage from our proposed secure scheme will be the same as those from the baseline dosage.

3.6.4 Discussion

In this section, we further discuss our proposed approach from both personalized medicine and security perspectives as well as suggesting future research directions.

Tradeoff Between Security and Efficiency

The Paillier cryptosystem achieves a higher security level with increasing security parameter k (i.e. key size). This, however, comes at the cost of decreased computation and communication efficiency. In our proposed scheme, we use $k = 1024$ bits which results in a ciphertext that is feasible to transmit as discussed in Section 3.6.2 while achieving a semantically secure data encryption [133]. However, very large values of k will (i) slow down homomorphic computation, and (ii) incur more data transmission overheads. For example, for values of k beyond 2048 bits ($k > 2048$ bits) the ciphertext pair is greater than 8192 bits since ciphertext space is bounded by n^2. For a patient with m data variables, his ciphertext pair will be greater than $8192m$ bits and consequently transmission cost will be lower bounded by $8192m$ bits (i.e. $> 8192m$ bits). Transmission cost is linear to mk (i.e. $O(mk)$) and real-world clinical settings should balance security and transmission cost by varying k.

Further Computation and Transmission Optimization

We have already established that our framework is (i) computationally efficient and acceptable for an outsourced model in Section 3.6.1, and (ii) data transmission is efficient and feasible in a real-world clinical settings in Section 3.6.2. However, further optimization can be applied to reduce the runtime and transmission overhead as well as data storage size on the cloud which might go a long way to save cloud computation

and storage cost. Techniques such as ciphertext packing [134] can be used to reduce the ciphertext size of encrypted patients genomic and clinical data.

3.7 Conclusion

In this chapter, we develop a novel secure framework to predict warfarin dosages without revealing patients sensitive information to address the current security and privacy challenges faced by dosing models in personalized medicine. To the best of our knowledge, our work is the first to propose a secure genotype-guided dosage prediction scheme while protecting patients privacy. We develop novel secure techniques to personalize warfarin dosage to patients' genetic variants using warfarin regression models, homomorphic computation and SMC protocols. We develop a new SMC comparison protocol to compare SNP states using homomorphic and blinding techniques. In addition, we propose a new encoding method for patients genetic variants, race and age to map their SNPs to integer values for dosage computation. We demonstrate through extensive experiments on a real-world patients dataset that our proposed scheme is secure, efficient, and practically feasible. Furthermore, warfarin dosages estimated by our scheme are similar to that of ground truth dosages. Our proposed framework can be extended in the future to incorporate ciphertext packing techniques to reduce the ciphertext size of encrypted patients genomic and clinical data to further optimize runtime and transmission cost.

Chapter 4: A Blockchain Implementation for Controlling Genomic Access and Variant Discovery Using Smart Contracts and Homomorphic Encryption

Genomic data repositories are rapidly growing due to the decline in the cost of DNA sequencing. This has increased the demand from stakeholders such as researchers to analyze these datasets to advance areas in biomedical research. Genomic datasets are mostly maintained by third-party direct-to-consumer (DTC) genomic companies who operate a business model of collecting DNA data from their customers and selling them to pharmaceutical companies. This puts the privacy of their customers at risk since each individual's human genome is unique. In addition, customers lose ownership of and access to their genomic data to DTCs and DTCs do not share profits from data sales with them. In this chapter, we propose a system based on blockchain technology and homomorphic computation to address the aforementioned problems. We use blockchain transactions and smart contracts to allow genomic data owners (DOs) to have control of their data and sell access to it, and homomorphic computation with secure two-party protocol to enable genomic data users (DUs) to run queries to securely discover DOs of interest. We further optimize the query response time by proposing an approach based on genomic data partitions, binary search trees and bloom filters to reduce the search space. We also propose a blockchain penalty mechanism to encourage parties to behave honestly to avoid malicious behaviors such as uploading fake or non-human genomic data. We conduct our experiments on real

NOTE: The content of this chapter has been submitted to *Future Generation Computer Systems.*
Mohammed Yakubu, A., & Chen, Y. P. P. A Blockchain Implementation for Controlling Genomic Access and Variant Discovery Using Smart Contracts and Homomorphic Encryption. *Future Generation Computer Systems.* (Under Review).

genomic datasets and demonstrate that our proposed scheme allows DOs to control access to their data and is feasible and efficient in terms of computation cost, query response time and scalability.

4.1 Introduction

Modern advancements in DNA sequencing techniques coupled with the decline in sequencing costs has resulted in the growth of genomic data collection. This data is vital in assisting researchers and clinicians to make discoveries and understand the structures and functionalities of biological data such as DNA, RNA and proteins. In the traditional genomic data sharing models, the avenues for collecting genomic data are via (i) non-profit organizations such as genomic data banks who collect genomic data to advance biomedical research. Examples are the 1000 genome project [3] and the personal genome project [135], and (ii) for-profit organizations such as direct-to-consumer (DTC) personal genomics companies who sell DNA sequencing and genetic testing (e.g., ancestry and genealogical services) to their customers. Both aforementioned approaches of acquiring genomic data raises the following concerns:
(1) The privacy and security of genomic data: the privacy of individuals who have contributed their DNA samples is at risk. This is because each individual's DNA is unique and contains highly sensitive information such as a predisposition to certain diseases etc., and when leaked might result in discrimination by employers and health insurance companies [109].
(2) Genomic data ownership and access: After individuals contribute DNA samples to a non-profit or for-profit organization, the ownership of the data is very much a grey area as individuals lose control of their data [136, 24]. Ideally, DNA sample donors should own and control their data. However, organizations entrusted with this data share access to other third parties without the consent of DNA donors.
(3) DNA donors do not profit from sharing their genomic data: for-profit DTCs monetize the genomic data of their customers. For instance, the pharmaceutical giant GlaxoSmithKline recently signed a $300m deal with 23andMe (a wellknown DTC company) to leverage their diverse genomic database and insights for the development

of new drugs [21, 22]. Customers do not benefit from the proceeds of such financial transactions.

A solution that addresses the above problems will motivate individuals to contribute their genetic information to genomic databases and repositories. A recent study showed that more than 50% of the population in the US would sell their genetic data for $95 with the right privacy measures in place [23]. Thus, there is a need for individuals who contribute their DNA samples to genomic databases to be able to securely control access to their data, while profiting from granting access to it. Blockchain is a distributed ledger technology (DLT) that stores transactional records in an immutable, tamper-resistant and transparent digital ledger. It is an emerging technology that has been applied in areas such as finance, health care, law enforcement and supply chains to transfer ownership of assets and value, and publicly record and verify transactions between parties without the need for a trusted entity [137]. We employ blockchain features (such as immutability etc.) together with fully homomorphic encryption to propose a novel framework to address the challenges of the traditional genomic data sharing model. Our framework (i) allows a DNA sample donor to own his genomic data and sell access to his data in return for digital currency, (ii) penalizes dishonest parties and reward honest parties in monetary value, and (iii) allows genomic data users (DUs) to discover the pseudo-anonymous blockchain addresses of genomic data owners (DOs) by running standard precise variant queries for genetic variants of interest. In addition, it is natural and fair for DUs such as clinical researchers to be able to download the DNA data to perform a broad range of studies on the data such as GWAS. Our framework integrates blockchain smart contracts to play a key role to enforce the fairness of genomic data exchange in that the DOs are paid and DUs get access to the genomic data.

Blockchain as an emerging technology has facilitated the development of frameworks to enable individuals to own and control access to high-value, privacy-sensitive data such as their genomic and medical health data. There are many studies on the application of the blockchain as an access control mechanism for patients to control their medical health data. The most recent studies in this direction have proposed

techniques to record medical health data access logs (i.e., for storage, querying, and retrieval) to the blockchain's immutable ledger to provide evidence of data ownership and usage [138, 139, 140]. Exchanging of patients medical health records between different health facilities have been a problem because health facilities isolate their data storage units from each other which causes delays in patients treatment plans. Recent studies have addressed this problem by employing blockchain smart contracts to develop token-based access mechanism to track and allow multiple health institutions to interoperate efficiently amongst themselves [141, 142]. Majority of research work on blockchain-based access control for sharing privacy-sensitive data focus on medical health data as stated above. However, there are limited studies that develop frameworks utilizing blockchain technology to control access and share genomic data as this is a new area of research. Current state-of-the-art techniques in this space have been proprietary genomic data sharing ecosystems such as Zenome project [143] and Nebula Genomics [144]. These systems provide a decentralized-blockchain platform for DOs to (i) store, manage, and control access to their data while maintaining privacy, and (ii) gain rewards for sharing their genomic data. However, the drawbacks of these techniques is that they lack the mechanism to reward and punish correct and dishonest behaviors respectively. Another recent study in this area develops blockchain access control smart contracts to enforce individuals consent and access policies over sharing their genomic data thereby eliminating the privacy risk of involving a DTC [145]. The limitation of [145] is that individuals are not compensated for sharing their data, and moreover query processing over large genomic datasets is expensive. Our proposed scheme addresses the aforementioned limitations of the state-of-the-art techniques that leverage blockchain features to control access and share genomic data. In summary, we make the following contributions in this chapter:

- We propose a novel genomic data sharing and access control framework utilizing blockchain functionalities driven by smart contracts to allow DOs to store, consent to share their data and sell access to DUs. In addition, we develop smart contracts to guarantee fairness between DOs and DUs to ensure DOs are paid and DUs get access to the data thereby eliminating the need of third party

DTCs.

- We develop a new penalty mechanism based on blockchain smart contracts and cryptocurrency to penalize dishonest parties and reward honest parties. Our approach ensures that DOs and DUs are incentivized to behave honestly and avoid misbehavior.

- We propose a novel partitioning technique to partition genomic data into non-overlapping blocks along chromosomal positions which we translate into a binary search tree. Our constructed partitions and binary search tree provides an efficient approach to run genomic discovery queries.

- We develop a new optimized approach to securely query DOs who meet a variant criteria from encrypted genomic datasets. Our technique utilizes homomorphic computation and SMC to securely match variants without revealing sensitive data. Our optimization reduces the query search space by orders of magnitude using our genomic data partitions, binary search tree and bloom filters.

- We implement our proposed secure blockchain-based genomic data sharing scheme on a private blockchain network and demonstrate using a real genomic dataset that our proposed scheme is efficient in terms of computation cost, query response time and scalability while protecting the privacy of DOs.

The rest of this chapter is organized as follows. In Section 4.2, we introduce the preliminaries and background required to understand our proposed scheme. The system model and security requirements are discussed in Section 4.3. In Section 4.4 we present our proposed framework for secure blockchain-based genomic data sharing with implementation details. The evaluation of security and the experimental setup and the performance analysis are presented in Sections 4.5 and 4.6 respectively. The chapter is concluded in Section 4.7.

4.2 Preliminaries and Background

In this section, we discuss the background on genomics basis, blockchain, fully homomorphic encryption and bloom filters which is needed to understand our proposed

framework.

4.2.1 Genomics Background

The human genome is the complete set of deoxyribonucleic acid (DNA) for humans which is encoded into two complementary DNA strands. It contains approximately 3 billion base pairs packed into 23 pairs of chromosomes. The DNA bases are adenine (A), thymine (T), guanine (G) and cytosine (C); A pairs with T and G pairs with C. About 99.9% of nucleotides are identical in any two individuals, the remaining 0.1% is due to genetic variations. The most common genetic variation occurs at a single position in a DNA sequence called single nucleotide polymorphism. There are several file formats and specifications for storing genetic sequence variations. A widely used format is the variant call format (VCF) [146] which consists of a header section and body. We refer readers to [146] for details of the VCF specifications. In our scheme the DO's genomic dataset is in VCF format.

4.2.2 Blockchain

Blockchain is a distributed ledger of immutable and tamper-resistant transactions stored on nodes of a decentralized peer-to-peer (P2P) network. Transactions in the network are collected and added to a block which is then validated by nodes using a consensus algorithm [137]. This validated block is then chained to the latest block in the blockchain by means of cryptographic hash pointers. The data structure of a block consists of a (i) block header which contains metadata such as the hash value of the current block and the previous block, the timestamp etc., and (ii) a block body which contains transaction data. There are three types of blockchain networks: private, public and consortium blockchain which have been applied in areas such as finance, healthcare etc., where parties agree to transact with each other without the need for trust. Terms and conditions of agreements among these parties are enforced by smart contracts (SC). A smart contract is a computer code that is immutably stored on a blockchain and automatically executes based on the established rules of a multiparty agreement. In this work we build a private blockchain network and deploy SCs on it

to enforce agreements between the DOs and the DUs.

4.2.3 Fully Homomorphic Encryption

A fully homomorphic encryption (FHE) scheme is a cryptographic construct that allows an unbounded number of operations on encrypted data [60]. First proposed in 2009 by Gentry, FHE was unpractical resulting in inefficient homomorphic computation protocols. However, recent improvements over the years have seen the construction of efficient variants of FHE (i.e. leveled-FHE and somewhat homomorphic schemes) based on ring learning with errors (RLWE) [60]. In our proposed scheme, we use Fan and Vercauteren (FV)'s cryptosystem [60] which is a practical and efficient implementation of FHE and provides security based on the hardness of the RLWE problem. The FV cryptosystem consists of the following algorithms.

Key generation

Let the polynomial rings $R_t = \mathbb{Z}_t[x]/(x^n+1)$ and $R_q = \mathbb{Z}_q[x]/(x^n+1)$ denote plaintext and ciphertext spaces respectively where n is a polynomial degree as a power of 2, t and q are the plaintext and coefficient modulus respectively, and χ is an error distribution over R_q. Then the secret key sk is uniformly sampled from R_q as $sk \xleftarrow{\$} R_q$, and randomly divided into two shares sk_1, sk_2 where $sk_1 \xleftarrow{\$} R_q$ and $sk_2 = sk - sk_1$. The public key is computed as $pk = (p_0, p_1) = ([-(a \times sk + e)] \bmod q, a)$ where $a \xleftarrow{\$} R_q$ and $e \leftarrow \chi$.

Encryption

For a message polynomial $m \in R_t$, let $\Delta = \lfloor q/t \rfloor$, $u \xleftarrow{\$} R_q$ and $(e_1, e_2) \leftarrow \chi$. Then the encryption function $E(.)$ is given as: $E(m) = c = \big((\Delta m + p_0 u + e_1) \bmod q, (p_1 u + e_2) \bmod q\big)$

Decryption

Given the ciphertext $c = (c_0 + c_1)$, the plaintext m can be recovered by the decryption function: $D(c) = \left\lfloor \frac{t}{q}(c_0 + c_1 \times sk) \bmod q \right\rceil \bmod t$

Partial Decryption

The ciphertext $c = (c_0 + c_1)$ can be partially decrypted with a share of the secret key sk_i ($i \in \{1, 2\}$) using the equation: $PD_{sk_i}(c) = (c'_0, c'_1) = (c_0 + c_1 \times sk_i, c_1) \bmod q$. The original FV cryptosystem does not have an algorithm for partial decryption so we use the technique proposed in [134] to partially decrypt the encrypted genomic variants in our work.

Homomorphic addition

Given two ciphertext c' and c'', the sum of the plaintext they encrypt can be computed as: $Sum(c', c'') = \big((c'_0 + c''_0) \bmod q, (c'_1 + c''_1) \bmod q\big)$.

Ciphertext packing

The FV cryptosystem supports the ciphertext packing technique which allows at most n messages to be packed into one plaintext polynomial and encrypted into one ciphertext. Homomorphic computation can then be applied on the ciphertext componentwise in a single instruction, multiple data (SIMD) manner. In its simplest form, for an n element message vector $\mathcal{M} = (m_1, m_2, ..., m_n)$ where $m_i \in \mathbb{Z}_t$ for $i \in (1, 2, ..., n)$, $\mathcal{M}$ can be packed into the plaintext polynomial $\mathcal{M}' = \sum_{i=1}^{n} m_{i-1}x^{i-1} = m_0 + mx + ... + m_n x^n$ and encrypted. We use this method to encrypt multiple genetic variants into one ciphertext. For brevity, in the rest of this chapter, we represent the encryption of a message m by $[m]$ and its partial decryption with $[m]^*$.

4.2.4 Bloom Filter

A bloom filter $\mathcal{BF}$ is a probabilistic space-efficient data structure used to test set membership. It is represented by an m bit array $\mathcal{BF} = (b_1, \ldots, b_m)$ with all bits initialized to 0. Given a set of t elements $S = \{e_1, \ldots, e_t\}$ and k independent hash functions $h_1, \ldots, h_k$ which maps to the range $[1, m]$, an element $e_i \in S$, for $i \in (1, \ldots, t)$ is added to the bloom filter by computing k hash values $h_1(e_i), \ldots, h_k(e_i)$ and setting their corresponding indexes in $\mathcal{BF}$ to 1 i.e. $\mathcal{BF}[h_j(e_i)] = 1$ for $1 \leq j \leq k$ and $1 \leq i \leq t$. To test for set membership, an element e is present in the bloom filter

if $\mathcal{BF}[h_j(e)] = 1$, else e is not in the set. We refer the reader to [147] for details. In our work, we employ bloom filters to improve the query efficiency of our genomic data discovery protocol by testing for membership of variants within a query sent from a DU.

4.3 Models and Assumptions

In this section, we discuss the system and security models with the assumptions we make in our proposed scheme.

4.3.1 System Model

Our system model as shown in Figure 4.1 is made up of various entities put together to provide a platform for DOs to own their data and sell access to it. Our system model comprising the following entities is depicted in Figure 4.1:

Genomic data owner (DO): DO is an individual who wants to share his genomic data, or an organization that owns genomic databanks and has sought consent from participants to share their data. DO can either submit his DNA sample (e.g. saliva), or an encrypted genomic data if already sequenced to the certified institution.

Genomic data user (DU): The DU is an entity such as a research facility that is granted access to a genomic data from a DO. Furthermore, DU pays a value in digital currency to the DO in exchange for data access.

Certified institution (CI): The CI sequences DNA, preprocesses and encrypts genomic data from DO prior to sending it to the off-chain storage node (SN) for storage.

Genomic data transaction party (DP): The DP is either a CI, DO or DU who registers on the blockchain network to perform transactions, write records and call smart contracts.

Off-chain storage node (SN): Data storage on the blockchain is expensive and comes with scalability issues. We move the data storage to an off-chain storage node for efficiency.

Computation server (CS): The CS provides online computation services. It collaborates with the SN to securely compute on the encrypted genomic data while

maintaining data privacy.

Blockchain network (BC): The BC is a decentralized peer-to-peer (P2P) network of nodes that provides an access control capability for DOs and a penalty mechanism by immutably recording transactions such as genomic data uploads and access request.

4.3.2 Security Model

In our security model, we assume DO, DU, SN, CS and BC are honest-but-curious parties in that they faithfully follow the protocol, but might try to infer sensitive information from other parties data. Furthermore, SN are CS are non-colluding parties. This means they do not collude to infer sensitive genomic data. The assumptions of honest-but-curious and non-colluding cloud servers have been applied in related work [145, 138, 139, 140] to model the behaviors of computing parties for information security. In real-world setting, competing cloud providers such as Amazon and Microsoft Azure do not collude as they are driven by business interest and commercial incentives of non-collusion [127, 145]. The CI is a semi-trusted entity who honestly sequences, validates, preprocesses and encrypts genomic data. It is important to note that DNA sequencing machines that generate encrypted data do not exist yet, so the CI would have access to the raw DNA data prior to encryption [98]. In a real-world setting, the role of CI can be played by the National Institutes of Health (NIH). While this requires DOs to have some level of trust in the CI to sequence their DNA, the blockchain as a trustless network plays a key role in our framework to (i) guarantee fairness between DO and DU (i.e. smart contracts ensures DO is paid and DU get data access), (ii) allow DO to control access to their data, and (iii) encourage honest behavior between parties using an immutable smart contract driven penalty mechanism. In addition to these assumptions, our security model meets the following requirements.

Data confidentiality: Any party other than the DO should not learn any sensitive information from the encrypted genomic data in storage and during computation.

Query privacy: The genomic discovery query should not leak any information about the variants being queried to unauthorized parties such as BC, SN and CS. In

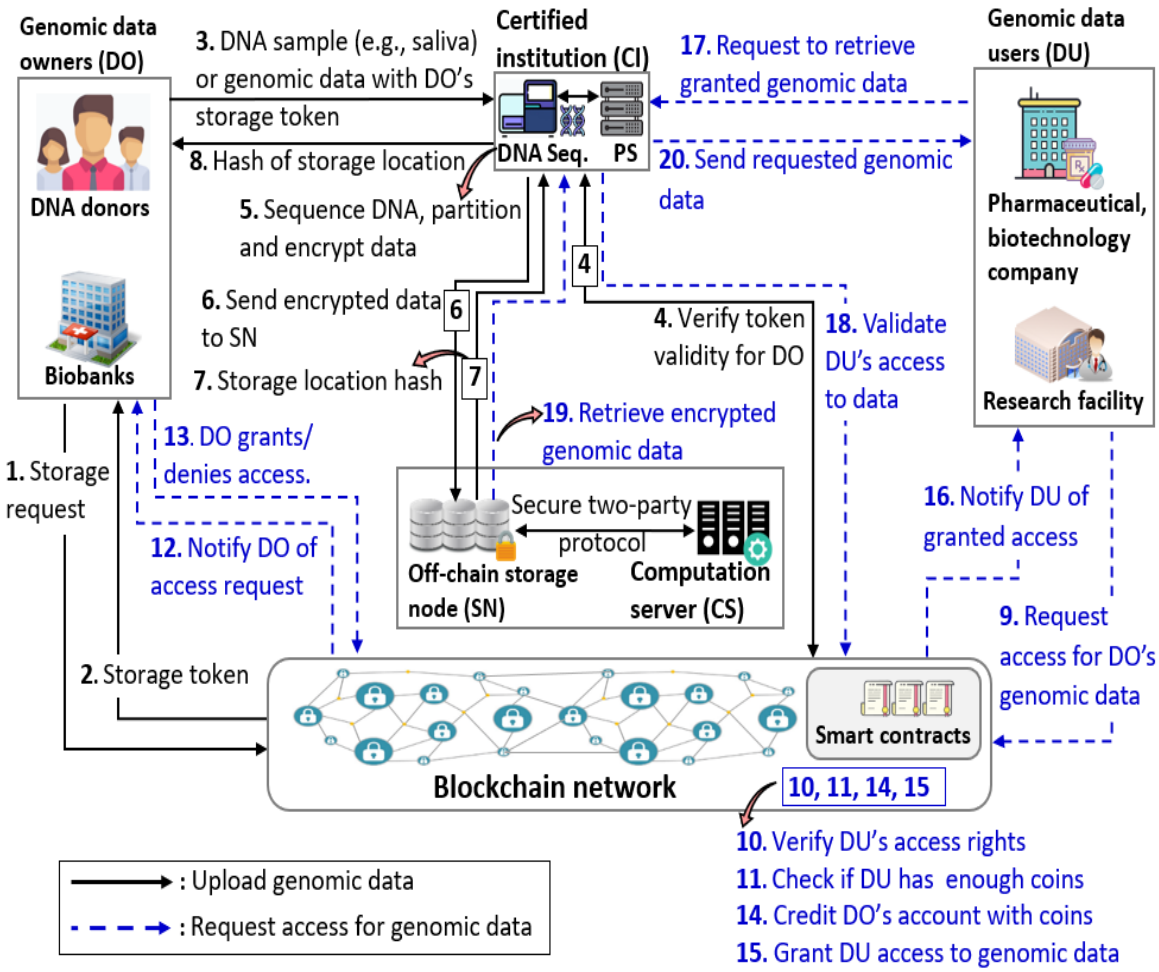


Figure 4.1: The system model for secure blockchain-based genomic data sharing. DOs send storage request to the BC for a token to upload genomic data. DO then sends his genomic data to CI. CI then validates DO, sequences DNA, preprocesses and encrypts the genomic data. The encrypted genomic data is sent to SN for storage. Finally, SN returns the hash of the storage location to CI which is then sent to DO. To request access to a DO's genomic data, DU sends an access request to BC, BC then validates DU and notifies DO of the request. DO then approves or denies the request and DU is notified by BC. DU finally retrieves the approved genomic data from SN via CI. ***Notes:*** **DNA Seq.**: DNA sequencer; **PS**: Processing server.

addition, the query result should not reveal the real identities of the DOs.

Fairness for genomic data exchange: Genomic data exchange between DO and DU should be fair meaning DO is paid and DU gets data access.

4.4 Our Proposed Scheme

In this section, we introduce and discuss our proof of concept implementation of our framework to allow DOs to share and control access to their data. We present our proposed framework in Figure 4.1 and its workflow of activities in Figure 4.2. We

denote DO's genomic data as $\mathcal{D} = \{RG, Ch, Pos, Ref, Alt\}$ where RG is the human reference genome[1] assembly build type, Ch is the chromosome number, Pos is the variant position, and Ref and Alt are the reference and alternate alleles respectively. We use $\mathcal{P}$ to denote phenotypic data (i.e. medical conditions) associated with $\mathcal{D}$. We propose encoding for (i) recent human reference genomes [148] as NCBI34 = 1, NCBI35 = 2, NCBI36 = 3, GRCh37 = 4 and GRCh38 = 5, (ii) DNA bases as $A = 1, G = 2, T = 3, C = 4, I = 5, D = 6, - = 0$. where I is an insertion, D is a deletion and $-$ is no-call, and (iii) medical conditions using the ICD-10 (international classification of diseases codes [149]). We propose to encode ICD-10 codes from alphanumeric to numeric. We convert the alphabet parts of the codes to two digits in increasing order from A to Z (i.e A=10, B=11, ..., Z=35), meaning ICD-10 codes from A00, A01, A03, ..., Z99 will convert to 1000, 1001, 1003, ..., 3599. For example, we encode breast cancer with ICD-10 of C50 to 1250.

In this section and the rest of the chapter we use a genotype to represent the concatenation of Ref and Alt i.e. $Ref||Alt$. We give an overview of our scheme with the following stages.

1. **Initialization**: A key generator (KG) initializes the system by generating the public key pk, the private key sk and random shares of sk i.e. sk_1 and sk_2. KG securely sends pk to CI, sk_1 to SN and sk_2 to CS.

2. **Registration of DO and DU onto BC**: As depicted in Figure 4.2, the DO and DU registers to join the private BC network. The BC then logs their registration transaction and generates a pseudo-anonymous address and a private key sk_{DP} for each user.

3. **Upload genomic data onto BC**: Next DO sends encrypted genomic data or DNA sample (e.g. saliva) along with phenotypic data to CI. CI sequences DNA, partitions data into blocks, hashes the variants of each block into a bloom filter and encrypts each block into one ciphertext. CI finally uploads encrypted data to SN. SN returns the hash of the storage location to CI. CI then invokes an SC to write the

[1]The human reference genome is an idealized genome assembled by scientists for the purposes of aligning and assembling an individual's genome sequence data. Recent assembly build types are NCBI34, NCBI35, NCBI36, GRCh37 and GRCh38 (i.e. national center for biotechnology information human genome build 34, 35, 36, and the genome reference consortium human genome build 37 and 38 respectively)

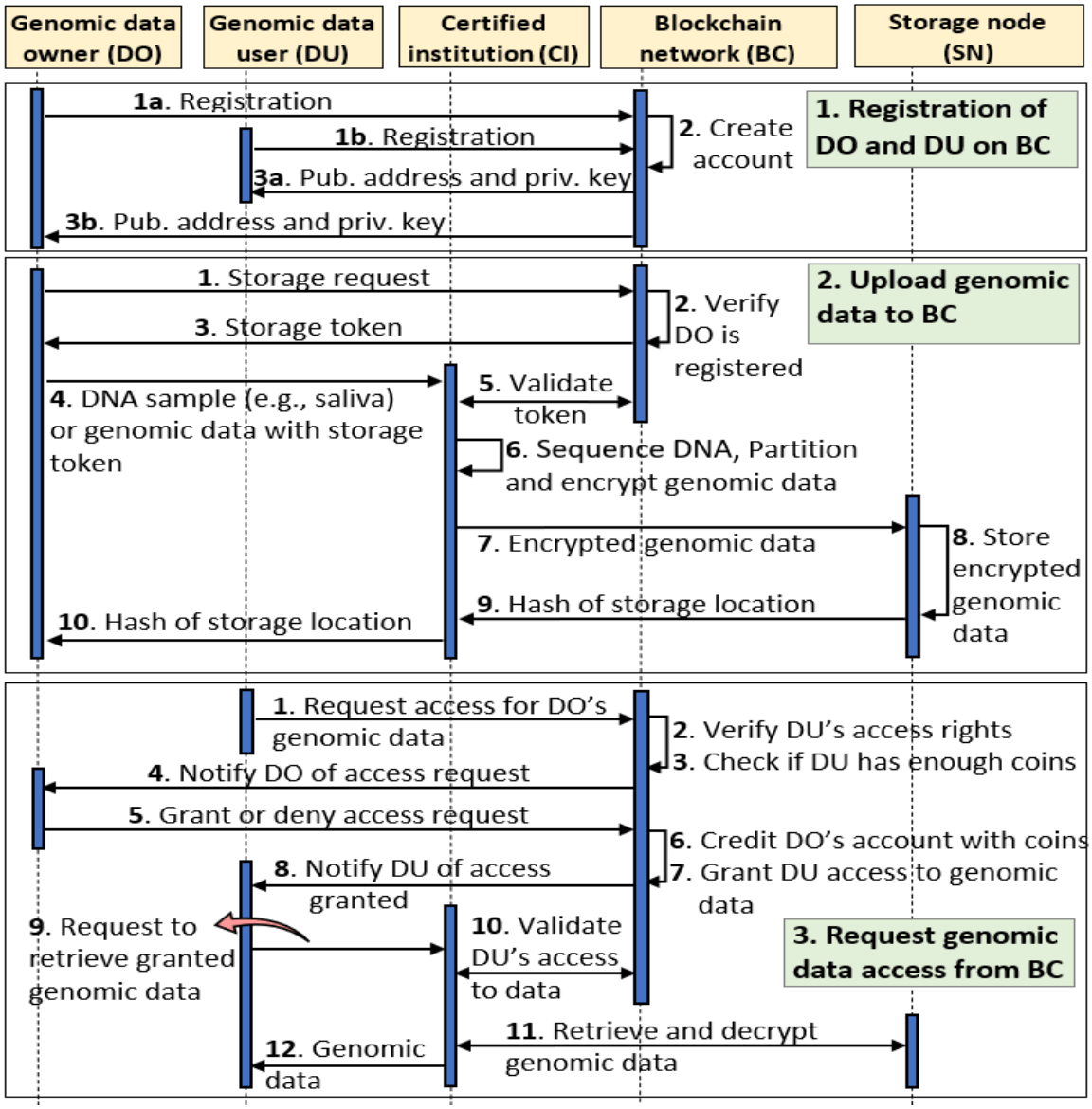


Figure 4.2: The workflow of processes involved in our proposed framework for secure blockchain-based genomic data sharing. It consist of the process steps for registration, genomic data upload from a DO and a request for genomic data from a DU.

data storage transaction into BC.

4. **Request genomic data access from BC**: DU sends a data access request with the address of interested DO to BC by invoking an SC. The SC verifies the DU's request with the access fee and notifies the DO of the request. DO then approves the request to redeem the access fee, and SC finally notifies DU to retrieve the data.

4.4.1 Preprocessing of Reference Genomes

The CI performs the initial preprocessing to partition the reference genomes into non-overlapping blocks, insert block keys into a binary search tree. This preprocessing is

a one-time operation performed during system setup.

Partitioning Reference Genome

Sequenced genomes are several positions long and searching for a variant position on a chromosome comes with a high computation cost. DNA is sequenced with respect to some reference genome which we propose to be used to speed up variant searching. In our work, we propose partitioning the reference genomes NCBI34, NCBI35, NCBI36, GRCh37 and GRCh38 into non-overlapping regions of chromosome positions. We then translate the partition blocks into a balanced binary tree. The goal of this is to reduce the search space of our secure genomic data discovery protocol (presented in Section 4.4.6). We accomplish this by grouping each reference genome into partitions of sizes l_i for $i \in \{1, 2, \ldots, p\}$ where p is the total number of partitions from chromosome 1 to chromosome MT (mitochondrial). We then compute an index $Pt_{i,indx}$ for each partition block Pt_i (i.e. the ith partition) as

$$Pt_{i,indx} = RG||Ch||i \tag{4.1}$$

where RG is the encoded value of the reference genome. Also we compute a value we call the partition code $Pt_{i,code}$ by generating a unique random number r_i to each partition i.e. $Pt_{i,code} = r_i$ for $r_i \in \mathbb{Z}_\mu$ where $\mathbb{Z}_\mu$ is variant position space (for $\mu >$ maximum variant position) and $r_1 \neq r_2 \neq \ldots \neq r_p$. By putting $Pt_{i,indx}$ and $Pt_{i,code}$ together, we construct a balanced binary search tree $\mathcal{BT}_{RG}$ for each reference genome as depicted in Figure 4.3 to enable fast search retrieval. We propose the attributes of each node of $\mathcal{BT}_{RG}$ as $N_{Pt_i} = \{key = Pt_{i,indx}, value = (Pt_{i,code}, Pt_{i,start}, Pt_{i,end}, l_i)\}$ where $Pt_{i,start}$ and $Pt_{i,end}$ is the partition's start and end positions respectively. For example partitioning reference genome GRCh38, the node attributes of partition 1 assuming block size is 5 is given as $N_{Pt_1} = \big\{key = (Pt_{1,indx} = 3811), value = (Pt_{1,code} = 82431381, Pt_{1,start} = 930188, Pt_{1,end} = 930336, l_1 = 5)\big\}$. CI generates one $\mathcal{BT}_{RG}$ per reference genome i.e. $\mathcal{BT}_{\Phi_{GRCh38}}$ for GRCh38, $\mathcal{BT}_{\Phi_{GRCh37}}$ for GRCh37, etc. It is important to note that partitioning the reference genomes and building $\mathcal{BT}_{RG}$ is a one-time operation and is updated as and when the reference genomes are revised by

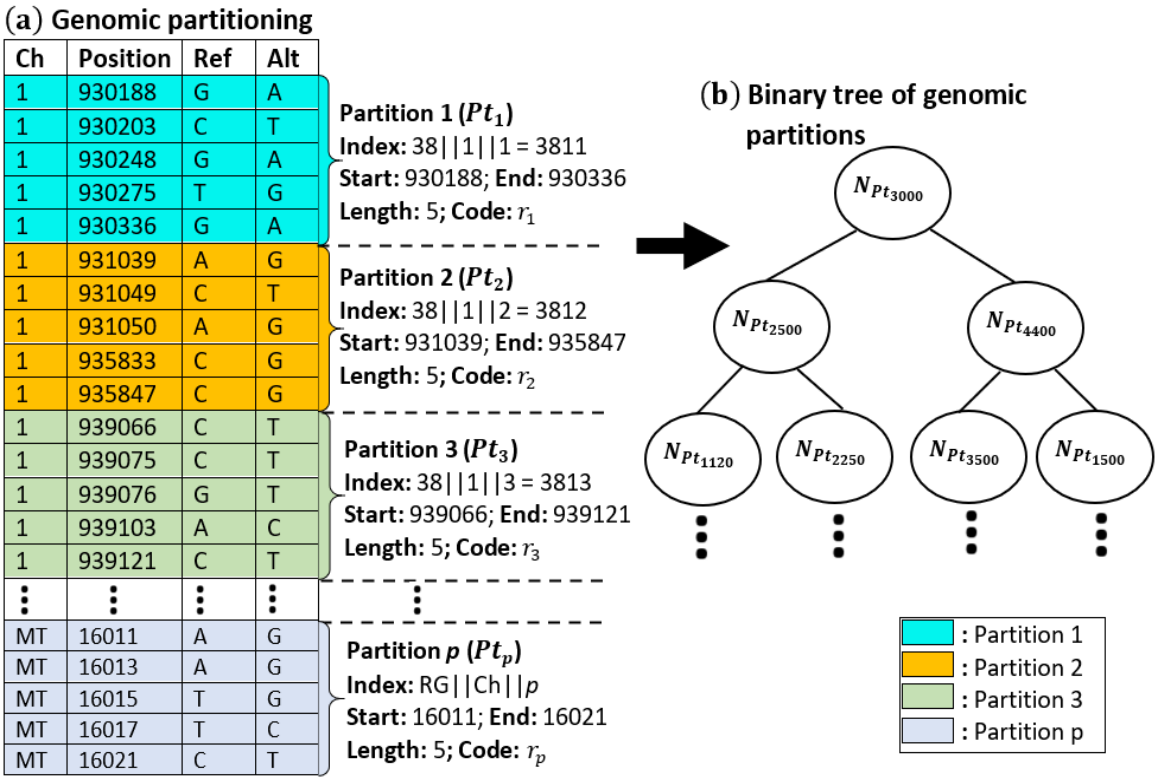


Figure 4.3: Partitioning reference genome (GRCh38) and constructing a balanced binary tree. (**a**) Proposed partitioning applied to the reference genome GRCh38. Genome positions are divided into non-overlapping blocks of size l (we use $l = 5$ for illustration purposes) from chromosome 1 to MT. (**b**) Balanced binary tree constructed from partitioned reference genome GRCh38 where each node is a genome partition. For brevity, we show nodes $N_{Pt_{3000}}, N_{Pt_{2500}}, ..., N_{Pt_{1500}}$ representing nodes of partitions $3000, 2500, ..., 1500$ respectively.

the genome reference consortium [148]

4.4.2 Blockchain-based Penalty Mechanism

We propose a new penalty technique based on blockchain to motivate a DP (i.e. DOs and DUs) to behave honestly. Our penalty technique rewards honest behavior and penalizes dishonest behavior using cryptographic coins i.e. $\$C$. Request and transactions from DPs to the BC are verifiable and our proposed framework detects dishonest behaviors such as (i) impersonating another user by using invalid signatures to learn sensitive information, and (ii) DO uploads fake or non-human genomic data. Next we show that each DP is incentivized to behave correctly with a high probability [150]. We denote the computation cost to carry out an honest and malicious behavior as ct_h and ct_m respectively. If a DP behaves honestly with probability Pr_h, then the probability for dishonest behavior is given as $Pr_m = Pr_c(1 - Pr_h)$; where Pr_c is the

probability that a DP's dishonest behavior will be checked. If ψ is the reward for correct behavior and f is the penalty for dishonest behavior, then the coins earned as a result of honest behavior is $\$C_{Pr_h} = \psi(1 - Pr_m) - fPr_m - ct_h$. In order to earn the maximum coins, a DP has to be completely honest i.e. $Pr_m = 0$ and $\$C_{Pr_h} = \psi - ct_h$. Next we show that the values of f, ψ and ct can be set to incentivize a DP to behave honestly.

Theorem 4 *A DP will behave honestly at least ϑ of the time if the penalty-to-reward ratio is set to $f/\psi > (1 - Pr_m)/Pr_m$, where $Pr_m = Pr_c(1 - \vartheta)$*

Proof 5 *Our proof shows that for any $Pr'_h < \vartheta$, the resulting coins earned are $\$C_{Pr'_h} < \$C_\vartheta$; where $\$C_{Pr'_h}$ and $\$C_\vartheta$ are the reward coins earned for behaving honestly with probabilities Pr'_h and ϑ respectively. Furthermore, we show that any DP who is dishonest with a probability of at least ϑ will earn negative coins meaning no incentive to act maliciously i.e. $\forall Pr'_h < \vartheta$, $\$C_{Pr'_h} < 0$. Recall that $\$C_{Pr'_h} = \psi(1 - Pr'_m) - fPr'_m - ct_{Pr'_h} < 0$. Now setting $f/\psi > 1/Pr'_m - 1$ guarantees that $\psi(1 - Pr'_m) - fPr'_m < 0$, and consequently $\$C_{Pr'_h} < 0$.*

In our penalty mechanism we propose that each DP commits to honest behavior by sending a commitment fee of $\$C_{commit}$ with a genomic data upload or access request to the BC. $\$C_{commit}$ is locked by an SC. In the event that a DP is dishonest, $\$C_{commit}$ is reallocated to honest DPs and the dishonest DP is deregistered from the platform to prevent future misbehavior. We propose **Algorithm 1** to penalize and reward DPs which takes as input the address of the dishonest DP $addr_m$ and his commitment $\$C_{commit}$.

4.4.3 Registration of Genomic Data Transaction Party (DP)

In our framework, a DP has to register onto the BC before using services such as uploading genomic data, browsing genomic data catalog etc. The registration phase adds a DP to the DP's list ($\mathcal{DPL}$) and generates public addresses for each DP which is used (i) for validation prior to any transaction, and (ii) to reward and penalize for dishonest behavior. The steps involved in our registration process are as follows:

Algorithm 1: *PenalizeRewardDP* (**PRD**) penalizes a dishonest DP by reallocating his commitment fee to honest DPs

Input: $addr_m, \$C_{commit}$
Output: *bool*

1. Get dishonest DP from registered DPs mapping object $\mathcal{DPL}$
   $\mathcal{DP}_m \leftarrow \mathcal{DPL}[addr_m]$
   if $\mathcal{DP}_m.id <= 0$ **then**
   throw
2. Increment malicious score of the dishonest DP $\mathcal{DP}_m$, and deregister $\mathcal{DP}_m$ from the blockchain
   $\mathcal{DP}_m.malscore \leftarrow \mathcal{DP}_m.malscore + 1$
   $\mathcal{DP}_m.unsubscribe \leftarrow true$
3. Iterate through $\mathcal{DPL}$ and get honest DPs set $\mathbb{H} \subseteq \mathcal{DPL}$ for $malscore = 0$
4. Reward honest DPs in the set $\mathbb{H}$
   $\$C_{payout} \leftarrow \$C_{commit}/count(\mathbb{H})$
   foreach *honest party* $\mathcal{DP}_h$ *in* $\mathbb{H}$ **do**
   $\mathcal{DP}_h.address.transfer(\$C_{payout})$ ▷ Transfer coins to $\mathcal{DP}_h$.
   $\mathcal{DP}_h.rewardcount$ ++ ▷ increment reward count.
5. **return** *true*

Step 1: To register, the DP sends a registration request to the BC. BC creates an account for the DP and generates a private key sk_{DP} and a public address i.e. $addr_{DP} \in \{addr_{DO}, addr_{DU}, addr_{CI}\}$ where $addr_{DO}$, $addr_{DU}$ and $addr_{CI}$ are public addresses for DO, DU and CI respectively. BC returns sk_{DP}, $addr_{DP}$ to the DP and invokes the SC's function **DPR** with inputs $addr_{DP}$ and $\mathcal{UT}$ (i.e user type $\mathcal{UT} \in \{DO, DU, CI\}$).

Step 2: **DPR** then adds $addr_{DP}$,$\mathcal{UT}$ to the registered DPs mapping $\mathcal{DPL}$[.] as detailed in **Algorithm 2**.

Step 3: Finally, the registration transaction is written to the BC ledger, and a success notification is returned.

We show the details of **DPR** in **Algorithm 2**.

4.4.4 Upload Genomic Data to the Blockchain Network

The DO uploads his genomic data $\mathcal{D}$ onto the platform to make it available to potential DUs. The DO sends his DNA sample or genomic data with phenotypic characteristics $\mathcal{P}$ to the CI. The CI then sequences, preprocesses and encrypts data before storing it on the SN. Below are the steps involved to upload a genomic data.

Step 1: DO first sends a storage request REQ_{store} to BC with a commitment fee $\$C_{commit}$ for honest behavior. i.e. $REQ_{store} = \{addr_{DO}, \$C_{commit}, \Gamma^{store}_{addr_{DO}}\}$, where

Algorithm 2: *RegisterDP* (**RDP**) registers a genomic data transaction party (DP) to the blockchain network

Input: $addr_{DP}$,$\mathcal{UT}$
Output: *bool*

1. Get DP's id for $addr_{DP}$ from registered DPs mapping object $\mathcal{DPL}$
   $id_{DP} \leftarrow \mathcal{DPL}[addr_{DP}].id$
2. If DP already exist and is registered on the BC network then throw
   if $id_{DP} > 0$ **then**
   ▷ DP already registered on BC as a genomic data transaction party
   throw
3. Increment registered DPs count and assign it as unique id for new DP
   $id_{DP} \leftarrow DPCount + 1$
4. Finally add new DP to registered DPs mapping object $\mathcal{DPL}$.
   $\mathcal{DPL}[addr_{DP}] \leftarrow \mathcal{DP}(id = id_{DP}, address = addr_{DP},$
   $userType = \mathcal{UT}, malscore = 0, rewardcount = 0)$
   return *true*

$\Gamma^{store}_{addr_{DO}}$ is $addr_{DO}$ signature of REQ_{store}. BC verifies DO's registration status and $\Gamma^{store}_{addr_{DO}}$. If $\Gamma^{store}_{addr_{DO}}$ is invalid, BC calls **Algorithm 1** to penalize $addr_{DO}$ by revoking $\$C_{commit}$ and rewarding honest DOs i.e. $\mathbf{PRD}(addr_{DO}, \$C_{commit})$. However if validation passes, BC generates a storage token for $addr_{DO}$ given as $tk_{store} = h(txNonce||addr_{DO}||t_i||t_e)$, where $txNonce$ is the BC transaction nonce for the request REQ_{store} and t_i and t_e are the issued and expiry time for the token. BC then sends tk_{store} to DO.

Step 2: Upon receiving tk_{store}, DO sends $\mathcal{D}, \mathcal{P}$ and tk_{store} to CI for sequencing, processing and encryption. CI verifies tk_{store} from BC. If tk_{store} is valid, CI sequences DO's DNA. However, if DO sent an already sequenced genomic data file instead a DNA sample, CI checks that the genomic data is not fake or non-human using well-known genetic verification methods presented in [143]. If the genomic data is fake or non-human, DO is punished by forfeiting $\$C_{commit}$ and the request is aborted i.e. $\mathbf{PRD}(addr_{DO}, \$C_{commit})$.

Step 3: CI then partitions DO's genomic data. Similar to partitioning the reference genome in Section 4.4.1, the data is divided into p partitions and the index $Pt_{i,indx}$ of each partition is computed using Equation 4.1. CI then uses the indexes $Pt_{i,indx}$ as search keys to quickly retrieve their corresponding codes $Pt_{i,code}$ from the binary tree $\mathcal{BT}_{RG}$ constructed in Section 4.4.1. CI then performs below:

1. First, CI packs the variants of each block Pt_i into a 2xk polynomial matrix $\mathcal{M}_i$ of positions and genotypes, and their corresponding number of DNA samples

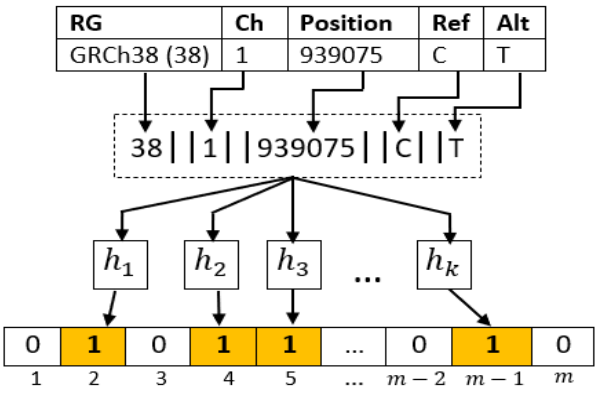


Figure 4.4: Bloom filter for encoding sample genomic variant at chromosome positions into an m-bit bloom filter for reference genome GRCh38.

(NSP) into a 1xk matrix $\mathcal{NS}_i$ as proposed in Figure 4.5. CI encrypts $\mathcal{M}_i$ to get $[\mathcal{M}_i]$. CI then builds the table $TB_{addr_{DO}}$ as shown in Figure 4.6 from $Pt_{i,code}$, $[\mathcal{M}_i]$ and $\mathcal{NS}_i$. We leave $\mathcal{NS}_i$ in plaintext as it is not sensitive.

2. Second, for each variant, CI hashes the string $\delta = RG||Ch||Pos||Ref||Alt$ k times into an m bit bloom filter as shown in Figure 4.4. We propose that CI generates and maintains one bloom filter per reference genome i.e. genomic data sequenced with reference to GRCh38 are encoded into BF_{GRCh38} etc. CI then packs each bit of BF_{GRCh38} and encrypts them into ciphertext $[BF_{GRCh38}]$.

Next, CI encodes the ICD codes of the medical conditions of DO present in $\mathcal{P}$ using the conversion technique we discussed earlier, and encrypts them into one ciphertext $[\mathcal{P}]$. In addition, CI encrypts the entire genomic data file and $\mathcal{P}$ with AES to obtain $F_{addr_{DO}}$. CI finally sends $TB_{addr_{DO}}$, $[\mathcal{P}]$, $F_{addr_{DO}}$ and $[BF_{GRCh38}]$ to SN for storage.

Step 4: SN receives the request from CI and updates Table $TB_{\Phi_{GRCh38}}$ with $TB_{addr_{DO}}$ as shown in Figure 4.6. Table $TB_{\Phi_{GRCh38}}$ contains encrypted partitions of variants of individuals whose DNA has been sequenced with reference to GRCh38. SN maintains one table per reference genome i.e. $TB_{\Phi_{GRCh38}}$ for GRCh38, $TB_{\Phi_{GRCh37}}$ for GRCh37, etc. SN also stores $[\mathcal{P}]$ in table TB_{ph}. Next SN replaces its local copy of $[BF_{GRCh38}]$ with the latest version from CI, and stores $F_{addr_{DO}}$ and computes a hash of the storage location as $loc_H = h(loc)$, where $h(.)$ is a hash function and loc is the storage path. Finally loc_H is sent to CI.

Step 5: CI receives loc_H as storage confirmation and encrypts it with AES to obtain $[loc_H]'$. CI then invokes the SC function **UGD** (**Algorithm 3**) with inputs $addr_{DO}, [loc_H]', \$C_{access}, tk_{store}$. **UGD** verifies if DO is registered and has a valid token

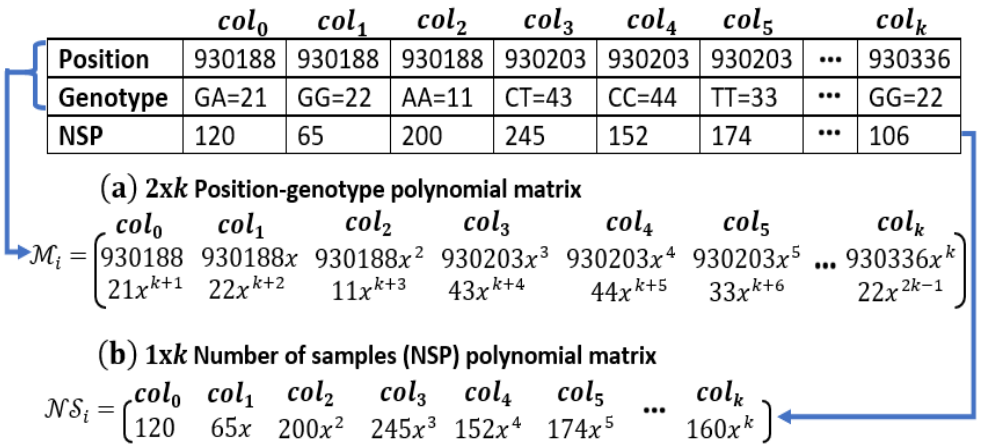


Figure 4.5: Proposed plaintext polynomial matrixes for genomic variants and the number of DNA samples. (**a**) 2xk position-genotype polynomial matrix $\mathcal{M}_i$ with chromosome positions on 1^{st} row and genotypes on the 2^{nd} row. (**b**) 1xk polynomial matrix $\mathcal{NS}_i$ for the number of DNA samples (NSP) corresponding to each position-genotype column index in $\mathcal{M}_i$.

tk_{store}. If valid, **UGD** assigns $addr_{DO}, [loc_H]'$ and $\$C_{access}$ to a data item variable for DO i.e., $dItem_{DO} \leftarrow \{Owner = addr_{DO}, Loc = [loc_H]', cost = \$C_{access}, \mathcal{AL} = null\}$, where $\$C_{access}$ is the cost to access the genomic data and $\mathcal{AL}$ is the access list for genomic data item $dItem_{DO}$. Finally, BC releases the withheld commitment fee $\$C_{commit}$ and refunds it back to DO.

Algorithm 3: *UploadGenomicData* (**UGD**) uploads a DOs genomic data onto the blockchain network

Input: $addr_{DO}, [loc_H]', \$C_{access}, tk_{store}$
Output: *bool*

1. Get id for $addr_{DO}$ from the registered genomic data transaction parties (DP) mapping
   $id_{DP} \leftarrow \mathcal{DPL}[addr_{DO}].id$
2. If DP does not exist or not registered in blockchain network then throw
   if $id_{DP} <= 0$ **then**
   throw
3. Validate storage token. If token is not valid or expired then throw
   if tk_{store} *is not valid or* $currentTime > t_e$ **then**
   throw
4. Get genome data item from genomic items mapping for $addr_{DO}$.
   $dItem \leftarrow \mathcal{G}[addr_{DO}]$
5. Assign uploaded genomic data properties $[loc_H]', \$C_{access}$ to $dItem$
   if $dItem.Owner == null$ **then**
   $dItem.Owner \leftarrow addr_{DO}$
   $dItem \leftarrow \{Owner = addr_{DO}, Loc = [loc_H]', cost = \$C_{access}, \mathcal{AL} = null\}$
7. **return** *true*

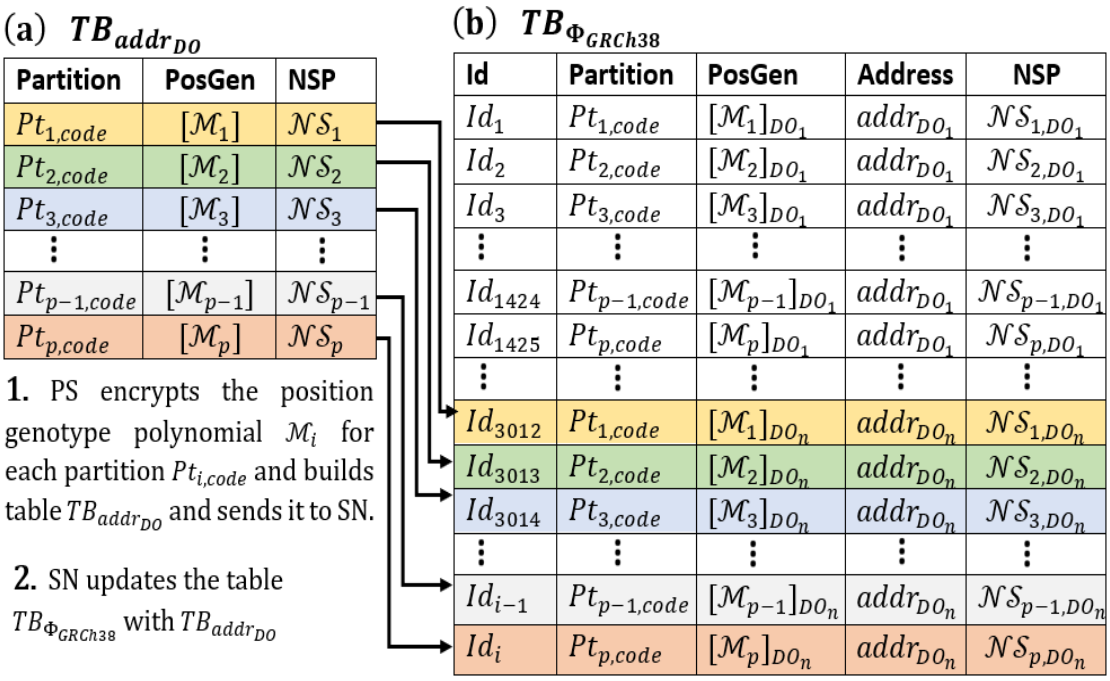


Figure 4.6: Genomic variant discovery tables. (**a**) Table $TB_{addr_{DO}}$ is build by CI from DO's genomic data. $TB_{addr_{DO}}$ contains the ciphertext $[\mathcal{M}_i]$ and the plaintext $\mathcal{NS}_i$ per partition block. (**b**) Table $TB_{\Phi_{GRCh38}}$ is stored on the SN and contains DOs encrypted variants per partition block. It is update with $TB_{addr_{DO}}$ each time a DO uploads his genomic data. ***Notes:*** **PosGen**: position-genotype; **NSP**: Number of DNA samples.

4.4.5 Request Genomic Data Access from the Blockchain

An access request for genomic data is initiated by a DU. Prior to sending an access request, the DU securely runs genomic data discovery queries to discover DOs with genetic variants of interest. The genomic data discovery process returns the addresses of DOs that match the query criteria. DUs follow the following steps to request genomic data access.

Step 1: DU sends an access request REQ_{access} with DO's address to BC by calling SC function **RGA** (i.e. **Algorithm 4**). We propose the fee $\$C_{\varrho} = \$C_{access} + \$C_{commit}$ to be sent along REQ_{access}, where $\$C_{access}$ is the genomic access fee and $\$C_{commit}$ is the commitment to honest behaviour. Access request is given as $REQ_{access} = \{addr_{DU}, addr_{DO_j}, \$C_{\varrho}, \Gamma^{ac}_{addr_{DU}}\}$, where $\Gamma^{ac}_{addr_{DU}}$ is DU's signature of REQ_{access}.

Step 2: BC receives REQ_{access} and verifies that (i) DU is registered and $\Gamma^{ac}_{addr_{DU}}$ is valid, and (ii) the fee is $\$C_{\varrho} \geq \$C_{access} + \$C_{commit}$. If $\Gamma^{ac}_{addr_{DU}}$ is invalid, BC calls **Algorithm 1** to apply the monetary penalty to $addr_{DU}$ i.e. $\mathbf{PRD}(addr_{DU}, \$C_{commit})$, and aborts the request and transfers $\$C_{access}$ back to the $addr_{DU}$. However, if checks

pass, BC sends notification NF_{access} to DOs and sets a cancellation timer t_c i.e. $NF_{access} \leftarrow \{txNonce, addr_{DO}, addr_{DU}, \$C_{access}, t_c\}$ where $txNonce$ is BC's transaction nonce of REQ_{access}. The DO has to respond to NF_{access} before t_c expires, else the SC will automatically terminate the request and refund $\$C_{\varrho}$ to $addr_{DU}$.

Step 3: DO responds to NF_{access} by sending request $REQ_{grant/deny}$ to either grant or deny access to $addr_{DU}$ i.e., $REQ_{grant/deny} = \{txNonce, addr_{DO}, addr_{DU}, \mathcal{AS}\}$ where $\mathcal{AS}$ is access states i.e., $AccReqSt$ $\{None = 0, Request = 1, Grant = 2, Deny = 3, Cancel = 4\}$

Step 4: The DO can either grant or deny access to his data by invoking SC function **GGA** (i.e. **Algorithm 5**). In the following, we explain the process to grant or deny genomic data access.

1. **Granting access:** If DO grants access, **GGA** credits his account with the access fee $\$C_{access}$, and generates an access token for DU to retrieve data i.e., $tk_{access} = h(txNonce||addr_{DO}||addr_{DU}||t_i||t_e)$, where t_i and t_e are the token's issued and expiry dates respectively. **GGA** then notifies (NF_{grant}) DU with token tk_{access} for genomic data retrieval. DU presents tk_{access} before expiry time t_e to retrieve genomic data. Finally, the commitment fee $\$C_{commit}$ is returned to DU for honest behavior.

2. **Denying access:** The DO may choose to deny access to his data. In such a case **GGA** refunds the fee $\$C_{\varrho}$ back to the DU and transaction is written to the blockchain.

3. **Canceling access:** The DU can cancel the request REQ_{access} before the DO acts on it. After cancellation, $\$C_{\varrho}$ is refunded back to the DU and transaction is recored to the blockchain.

4.4.6 Genomic Data Discovery Based on Variants of Interest

In our framework, genomic data discovery is a process where a DU learns of the availability of genomic datasets with certain phenotypic traits or genetic variants of interest by running standard precise variant (SPV) queries. Standard precise variant

Algorithm 4: *RequestGenomicAccess* (**RGA**) processes access request for a DO's genomic data from a DU.

Input: $addr_{DO}, addr_{DU}, \$C_{\varrho}, \Gamma^{ac}_{addr_{DU}}$
Output: $\perp$

1. Require that $addr_{DU}$ is not the owner of the genomic data
   if $addr_{DO} == addr_{DU}$ **then**
   throw
2. Verify DU's signature. If it's not valid, apply monetary penalty
   if $\left(\Gamma^{ac}_{addr_{DU}}$ *is invalid* $\right)$ **then**
   PRD$(addr_{DU}, \$C_{commit})$ ▷ Penalize $addr_{DU}$
   Abort Request
3. Get genome item from genomic items mapping for $addr_{DO}$.
   $dItem \leftarrow \mathcal{G}[addr_{DO}]$
4. Get and update access list item of $dItem$ from $addr_{DO}$ to $addr_{DU}$.
   if $dItem.Owner$! $= null$ **then**
   $\mathcal{AL} \leftarrow dItem[addr_{DU}]$
   if $\mathcal{AL}.\mathcal{AS}$! $=$ *Request* || $\mathcal{AL}.\mathcal{AS}$! $=$ *Grant* **then**
   if $\$C_{\varrho} < (dItem.cost + deposit)$ **then**
   ▷ check if DU has enough coins for the transaction
   throw
   $\mathcal{AL}.\mathcal{AS} \leftarrow AccReqSt.Request$
   $\mathcal{AL}.CommitFee \leftarrow deposit$
   $\mathcal{AL}.AccessFee \leftarrow \$C_{access} = (\$C_{\varrho} - deposit)$
   $NF_{access,t_c} \leftarrow \{txNonce, addr_{DO}, addr_{DU},$
   $\$C_{access}\}$ ▷ Notify $addr_{DO}$ of requested access

query is used to identity the presence of genetic variants within genomic datasets by querying for the variants reference genome, start position, reference and alternate alleles [151]. Querying for DOs by certain phenotypic traits such as diseases is trivial so in this section we focus on securely running queries to match genetic variants of interest. We propose a novel technique which enables the DU to securely query the SN for genomic datasets that contains variants of interest. Our secure approach meets the requirement where (i) the variants queried by the DU are kept private and are not disclosed to unauthorized parties such as BC, SN and CS, and (ii) the real identities of the DO's to any unauthorized parties such as SN, CS and BC nodes. Next we discuss the process to discover a genomic variant.

Formulating query at the DU

A DU who wishes to discover the presence of a certain genetic variant (e.g., for the breast cancer variant on BRCA1 gene i.e. $RG = GRCh38, Ch = 17, Pos = 43088733, Ref = G, Alt = A$) first formulates a query. We propose the format for a genomic data discovery query $\mathcal{Q}$ as: $\mathcal{Q} = \{RG, Ch, Pos, Ref||Alt\}$. DU then per-

Algorithm 5: *GrantGenomicAccess* (**GGA**) grants or denies access to a DO's genomic data

Input: $addr_{DO}, addr_{DU}, \mathcal{AS}$
Output: $\perp$

1. Require that $addr_{DU}$ is not the owner of the genomic data
   if $addr_{DO} == addr_{DU}$ **then**
      throw
2. Get genome item from genomic items mapping for $addr_{DO}$.
   $dItem \leftarrow \mathcal{G}[addr_{DO}]$
3. Update access list item of $dItem$ from $addr_{DO}$ to $addr_{DU}$.
   if $dItem.Owner\ != null$ **then**
      $\mathcal{AL} \leftarrow dItem[addr_{DU}]$
      if $\mathcal{AL}.\mathcal{AS}\ ==\ Request$ **then**
         if $\mathcal{AS} == AccReqSt.Grant$ **then**
            $addr_{DO}.transfer(\mathcal{AL}.AccessFee)$ ▷ Transfer access fee to DO.
            $tk_{access} \leftarrow h(txNonce||addr_{DO}||addr_{DU}||t_i||t_e)$ $\mathcal{AL} \leftarrow$ update access list to grant
            $NF_{grant} \leftarrow \{txNonce, tk_{access}, t_i, t_e,$ $AccReqSt.grant\}$ ▷ Notify $addr_{DU}$ of access grant.
         else if $\mathcal{AS} == AccReqState.Deny$ **then**
            $\$C_\varrho \leftarrow \mathcal{AL}.commitfee + \mathcal{AL}.accessfee\ addr_{DU}.transfer(\$C_\varrho)$ ▷ Refund $\$C_\varrho$ to DU.
            $\mathcal{AL} \leftarrow$ update access list to deny
            $NF_{deny} \leftarrow \{txNonce, AccReqSt.Deny\}$ ▷ Notify $addr_{DU}$ of denied request
         else if $\mathcal{AS} == AccReqSt.Cancel$ **then**
            $\$C_\varrho \leftarrow \mathcal{AL}.commitfee + \mathcal{AL}.accessfee\ addr_{DU}.transfer(\$C_\varrho)$ ▷ Refund $\$C_\varrho$ to DU.
            $\mathcal{AL} \leftarrow$ update access list to cancel
         else
            ▷ Do nothing.

forms below:

1. Generates and encrypts a probe filter BF_q: The goal of the probe filter BF_q is to securely test for the membership of the query variants within the bloom filter $[BF_{GRCh38}]$ on SN. DU generates an m-bit bloom filter BF_q and hashes Q with k hash functions into the range of BF_q as depicted in Figure 4.7. DU then packs each bit of BF_q and encrypts it into one ciphertext $[BF_q]$ with the public key pk .

2. Encryption of the query: DU then formulates the polynomial matrix $\mathcal{M}_q$ using the same technique presented in Figure 4.5 with the query position and genotype at index col_q and all other indexes set to 0. DU then uses the public key pk to encrypt $\mathcal{M}_q$ to get $[\mathcal{M}_q]$. Finally, DU sends $\mathcal{Q}' = \{RG, Ch, Pos, [\mathcal{M}_q], [BF_q], \Gamma_{DU,\mathcal{Q}}\}$ to CI; where $\Gamma_{DU,\mathcal{Q}}$ is DU's digital signature of $\mathcal{Q}$.

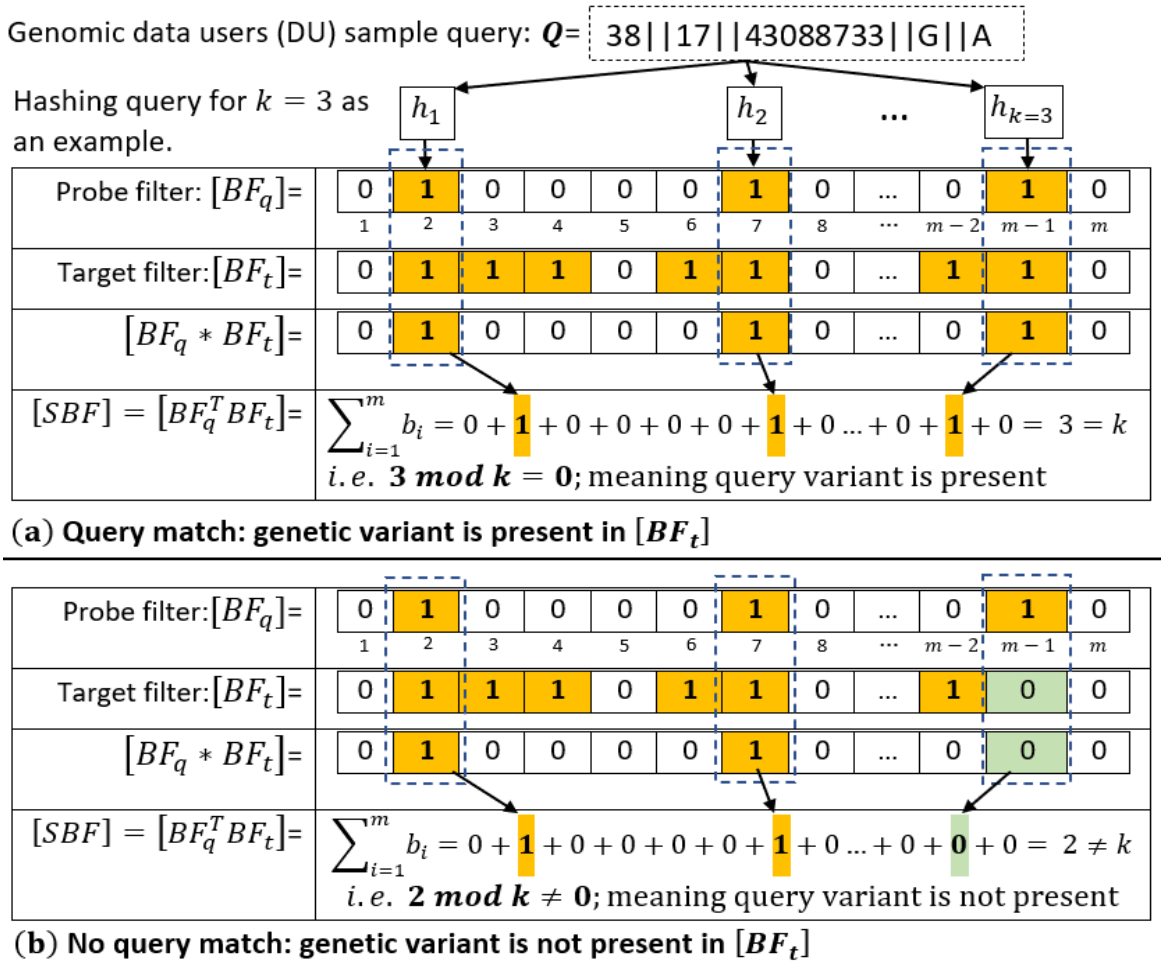


Figure 4.7: Secure bloom filter membership testing for queried genetic variants from the inner dot product $[BF_q^T BF_t]$. (a) Query match: genetic variant is present in BF_t with a high probability (b) No query match: genetic variant is not present in BF_t

Processing query at the CI

At CI, $\Gamma_{DU,\mathcal{Q}}$ is first validated. If valid, CI performs below:

Search optimization: CI reduces the search space by determining the partition block code $Pt_{i,code}$ to which the query paramaters (RG, Ch, Pos) belongs. CI computes the partition index $Pt_{i,indx}$ using Equation 4.1, and retrieves $Pt_{i,code}$ from the binary search tree $\mathcal{BT}_{RG}$ for $key = Pt_{i,indx}$. We denote the partition block to which the query variant belongs as $Pt_{q,code} = Pt_{i,code}$. Finally, CI sends $\mathcal{Q}'' = \{RG, Pt_{q,code}, [\mathcal{M}_q], [BF_q]\}$ to SN for secure processing.

Processing query at the SN

We process the query $\mathcal{Q}''$ at SN securely without revealing any sensitive information about the genetic variants, the real identities of the DO and DU or the query variant

index positions within the bloom filters. We propose an efficient SMC based protocol using homomorphic computation and blinding techniques between SN and CS. SN securely checks for the presence of the queried variants in the bloom filter. If present, SN proceeds to query the database. The process is as follows:

1. Test for the presence of query variants in bloom filter: First, SN securely test for the membership of genetic variants from the intersection of the probe filter $[BF_q]$ and the target filter $[BF_t]$. The target bloom filter $[BF_t]$ is the latest version of the filter $[BF_{GRCh38}]$ updated per data upload from CI as presented in Section 4.4.4. As shown in Figure 4.7, we propose a new technique to securely test variant membership in a bloom filter. We propose SN computes the secure dot product between $[BF_q]$ and $[BF_t]$ i.e. $[SF] = [BF_q^T BF_t] = [\sum_{i=1}^{m} b_{q_i} b_{t_i} = b_{q_1} b_{t_1} + ... + b_{q_m} b_{t_m}]$ where BF_q^T is the transpose vector of bits in BF_q, b_{q_i} and b_{t_i} are bits at the ith position in $[BF_q^T]$ and $[BF_t]$ respectively. SN then partially decrypts $[SF]$ with sk_1 to obtain $[SF]^*$ and send it to CS. CS decrypts $[SF]^*$ to obtain SF. It is important to note that SF contains only the summation of the bits at the query variant's hashed index positions in BF_q and BF_t for which there is a match and as such does not reveal information about the variants or their hashed index positions. Now CS checks for the presence of the query variants by evaluating SF. If $(SF \bmod k = 0)$; where k is the number of hash functions, then there is high probability that query variants are present in the bloom filter and hence the database, else there is no query match. CS returns $true$ or $false$ to SN. If SN receives a $false$, it delivers a "no query match" to CI for onward delivery to DU, else it proceeds to query the database.

2. Query database: SN retrieves the records $rs(Id_j, [\mathcal{M}_j])$ from table $TB_{\Phi_{GRCh38}}$ for partition $Pt_{q,code}$. SN then computes the homomorphic difference between $[\mathcal{M}_q]$ and $[\mathcal{M}_j]$, and blinds the encrypted result with a randomly generated number $r_q \in \mathbb{Z}_q$ to get $[\Upsilon_j]$. SN then partially decrypts $[\Upsilon_j]$ with sk_1 to obtain $[\Upsilon_j]^*$, and sends $(Id_j, [\Upsilon_j]^*)$ to CS to initiate a secure comparison protocol. CS decrypts $[\Upsilon_j]^*$ with sk_2 to obtain the polynomial Υ_j, and returns the set of Ids $\mathcal{I}$ to SN for which there is a query match i.e. position = 0 and genotype variant = 0. With ids $\mathcal{I}$, SN retrieves the address set $rs(\Omega)$ of DO addresses and number of DNA samples from the table

Algorithm 6: *DiscoverGenomicData* (**DGD**) processes queries from DU for DOs with genetic variants of interest

Input: DU: $addr_{DU}, \mathcal{Q}$; CI: $\mathcal{Q}'', pk$; SN: sk_1; CS: sk_2
Output: $rs(\Omega)$

1. **DU:** (a) Formulate and encrypt query $\mathcal{Q}$ and probe filter BF_q.
   $\mathcal{Q}' \leftarrow \{RG, Ch, Pos, [\mathcal{M}_q], [BF_q], \Gamma_{DU,\mathcal{Q}}\}$
   (b) Send $\mathcal{Q}'$ and $addr_{DU}$ to CI.
2. **CI:** (a) Optimize search space. Retrieve $Pt_{i,code}$ from $\mathcal{BT}_{RG}$.
   $Pt_{q,code} \leftarrow Pt_{i,code} = \mathcal{BT}_{RG}(key = Pt_{i,indx})$
   (b) Send $\mathcal{Q}'' \leftarrow \{RG, Pt_{q,code}, [\mathcal{M}_q], [BF_q]\}$ to SN.
3. **SN:** (a) Test variants presence in filters. Compute dot product of $[BF_q]$ and $[BF_t]$.
   $[BF_q^T] \leftarrow$ **Transpose**$([BF_q])$; $[SF] \leftarrow [BF_q^T BF_t]$;
   (b) Partially decrypts $[SF]$ to obtain $[SF]^*$ and send to CS.
4. **CS:** (a) Decrypts $[SF]^*$ with sk_2. i.e. $SF \leftarrow D_{sk_2}([SF]^*)$.
   (b) Variant is present if SF is a factor of k
   $isPresent \leftarrow false$
   if $(SF \ mod \ k = 0)$ **then**
   $isPresent \leftarrow true$
   (c) Return $isPresent$ to SN.
5. **SN:** (a) Return "no match" to CI if variant is absent.
   (b) Query database if variant is present. Retrieve records rs from table.
   $rs(Id_j, [\mathcal{M}_j]) \leftarrow$ **select** $Id, PosGen$ **from** $TB_{\Phi_{GRCh38}}$ **where** $Partition = Pt_{q,code}$
   (c) Securely subtract $[\mathcal{M}_q]$ from $[\mathcal{M}_j]$, and randomly blind the result with $r_q \in \mathbb{Z}_q$. i.e.
   $[\Upsilon_j] = r_q \times \Big([\mathcal{M}_q] - [\mathcal{M}_j]\Big)$
   (d) Partially decrypts $[\Upsilon_j]$ to get $[\Upsilon_j]^*$ and send $(Id_j, [\Upsilon_j]^*)$ to CS.
4. **CS:** (a) Decrypts $[\Upsilon_j]^*$ with sk_2. i.e. $\Upsilon_j \leftarrow D_{sk_2}([\Upsilon_j]^*)$.
   (b) Get ids set $\mathcal{I}$ for which there is a query match in polynomial Υ_j
   for $j = 0$ *to number-of-cols in* Υ **do**
   if *position* $= 0$ *and genotype* $= 0$ **then**
   add Id_j to $\mathcal{I}$
   (c) Return set of ids $\mathcal{I}$ to SN.
4. **SN:** (a) Retrieve addresses and number of samples from $TB_{\xi_{GRCh38}}$.
   $rs(\Omega) \leftarrow$ **select** $Address, NSP$ **from** $TB_{\xi_{GRCh38}}$
   where $Partition = Pt_{q,code}$ **and** Id **in** $\mathcal{I}$
   (b) Send $rs(\Omega)$ to CI for onward submission to the DU.

$TB_{\Phi_{GRCh38}}$ as detailed in **Algorithm 6**. Finally, SN returns $rs(\Omega)$ to CI, and CI in turn forwards it to the DU and the transaction is recorded in BC for auditing purposes. We remark that at the end of this protocol, the SN and CS do not learn any sensitive variant information about the DO or from the DU's query.

4.5 Security Analysis

In this section, we analyze the security of our proposed protocols **DGD**, **DPR**, **RGA** and **GGA** and our proposed scheme as a whole. We then demonstrate that our scheme meets the security requirements discussed in Section 4.3.2.

4.5.1 Security of Our Proposed Protocols

In order to analyze the security of our proposed protocols we present a formal definition of security under non-colluding semi-honest adversaries as follows.

Definition 2 (security in the non-colluding semi-honest model [15]) *Let Ψ be a protocol between n parties $p_1, p_2, \ldots, p_n$ with I_{p_i} and O_{p_i} as a set of p_i's input and outputs respectively. Let $\mathcal{K}(.)$ be a deterministic function and $VW_{p_i}^{\Psi}$ represent party p_i's execution view of protocol Ψ. For each party p_i, Ψ securely computes $\mathcal{K}(.)$ under the semi-honest model if there exist a polynomial-time algorithm $SM_{p_i}^{\Psi}$ that simulates its view such that: $SM_{p_i}^{\Psi}(I_{p_i}, \mathcal{K}(.)) \overset{c}{\approx} VW_{p_i}^{\Psi}(I_{p_i}, O_{p_i})$, where $\overset{c}{\approx}$ denotes computationally indistinguishable.*

Theorem 5 *The **DGD** protocol described in Section 4.4.6 securely discovers the presence of queried genomic variants over the encrypted genomic datasets in the presence of semi-honest (non-colluding) adversaries.*

Proof 6 *The nodes involved in our DGD protocol are CI, SN and CS. We omit CI from our proof since after sequencing the DNA, it has access to the raw data before encryption. In our proof we show that the views of SN (VW_{SN}^{DGD}) and CS (VW_{CS}^{DGD}) are computationally indistinguishable from their simulated images SM_{SN}^{DGD} and SM_{CS}^{DGD} respectively. The view of SN from the DGD protocol is $VW_{SN}^{DGD} = \{\mathcal{Q}'', rs\}$. This gives $VW_{SN}^{DGD} = \{\mathcal{Q}'' = [\mathcal{M}_q], rs = [\mathcal{M}_j)\}$. Now simulating an image of VW_{SN}^{DGD} will give $SM_{SN}^{DGD} = \{\widehat{\mathcal{Q}''} = [\widehat{\mathcal{M}_q}], \widehat{rs} = [\widehat{\mathcal{M}_j}]\}$; where $[\widehat{\mathcal{M}_q}]$ and $[\widehat{\mathcal{M}_j}]$ are randomly generated from $\mathbb{Z}_q$. Considering the fact that FV cryptosystem used to encrypt the elements of VW_{SN}^{DGD} and SM_{SN}^{DGD} ensures indistinguishability under a chosen-plaintext attack (IND-CPA) if the RLWE problem is hard, we can conclude that VW_{SN}^{DGD} is computationally indistinguishable from SM_{SN}^{DGD}. Likewise the view of CS is $VW_{CS}^{DGD} = \{[\Upsilon]^*\}$ and its simulated image is $SM_{CS}^{DGD} = \{[\hat{\Upsilon}]^*\}$; where $[\hat{\Upsilon}]^*$ is randomly generated from $\mathbb{Z}_q$. We conclude that since $[\Upsilon]^*$ and $[\hat{\Upsilon}]^*$ are encrypted with the FV cryptosystem which is IND-CPA secure, VW_{CS}^{DGD} is computationally indistinguishable from SM_{CS}^{DGD}.*

Next we evaluate the security of protocols **DPR**, **RGA** and **GGA**. Here our analysis is two-fold: (i) the protocols **DPR**, **RGA** and **GGA** do not process sensitive

information such as genomic data, instead they take the BC addresses of DO ($addr_{DO}$) and DU ($addr_{DU}$) as inputs to update the state of the SC, and (ii) the addresses $addr_{DO}$ and $addr_{DU}$ and the transactions they write to the BC are pseudo-anonymous which means they do not connect to the real identities of DPs. Therefore, we can conclude that protocols **DPR**, **RGA** and **GGA** do not leak sensitive information about the DPs.

4.5.2 Security of Our Proposed Scheme

The security of our proposed scheme is based on (i) the security of our proposed protocols **DGD**, **DPR**, **RGA** and **GGA** which we have already proven to be secure, and (ii) the security of DP accounts on the BC which is guaranteed by public key cryptography (PKC). In addition, our proposed scheme meets the security goals presented in Section 4.3.2 as follows

Genomic data confidentiality

The confidentiality of DO's genomic data is met since (i) genomic variants are encrypted with FV encryption which provides IND-CPA security, and (ii) the genomic data file (i.e. VCF) is encrypted with AES. In addition, secure computation for **DGD** is performed based on additive homomorphic properties and leaks no sensitive information as already proven by Theorem 5.

Query privacy

The genetic variants of the query and the probe bloom filter sent from DU are encrypted by the FV cryptosystem. In addition, the SMC computation between SN and CS does not reveal any sensitive information about the variants. Hence we conclude that the privacy of the query is guaranteed as it does not reveal sensitive genomic variants to unauthorized parties such as BC, SN and CS.

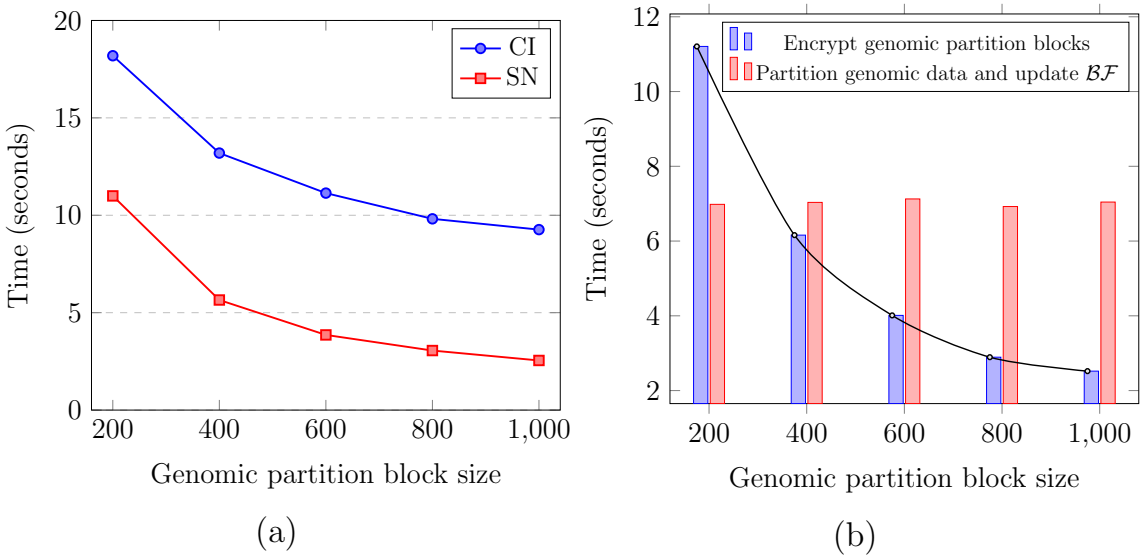


Figure 4.8: Computation time (seconds) to upload data on the parties by varying genomic partition block sizes (PB) 200, 400, 600, 800 and 1000. **(a)** Runtime (seconds) on the CI and SN for processing and uploading genomic data. **(b)** Runtime (seconds) on CI to partition genomic data into blocks, update the bloom filter ($\mathcal{BF}$) and encrypt the partition blocks.

4.6 Implementation and Evaluation

In this section, we implement our scheme and analyze its performance. We implement our framework in C# and run the experiments on a PC with 2.60 GHz processor (4 cores) and 16 GB memory. First, we build a private blockchain network with Ethereum (using geth) with 10 nodes. We deploy our smart contracts on system setup. For the FV cryptosystem, we use the Microsoft SEAL library [152] which is an open-source library dedicated to ideal lattice cryptography. We set the FV cryptosystem parameters to obtain a security level of 128 bits i.e. polynomial modulus degree to 2048 and plaintext and ciphertext modulus to 20 bits and 54 bits respectively. We run our experiments on genomic datasets from the personal genome project (PGP) [135] which is a public genetic repository consisting of 1572 participants whose DNA has been sequenced by DTCs such as 23andMe, family tree DNA etc. We then assessed the performance of our proposed scheme in terms of computation cost, query response time and scalability.

4.6.1 Computational Cost to Upload Genomic Data

Here we evaluate and discuss the computational cost in terms of computation runtime to upload genomic data in our proposed scheme. The process involves parties DO, CI and SN. We omit DO from our evaluation as the only task it performs is to send genomic data to CI for processing and hence incurs a negligible runtime cost. In our proposed scheme the processing time to upload genomic data is mainly dependent on the number of genomic partition blocks to process. Hence, we evaluate the runtime by varying partition block sizes in steps of 200 from 200 to 1000 (i.e. PB $= 200, 400, ..., 1000$). We present the average runtime in seconds (over 10 runs) on parties CI and SN by varying PB in Figure 4.8a. It is evident from Figure 4.8a that the runtime on parties CI and SN decreases with increasing PB. This is because as PB grows more genomic variants are packed into a single polynomial thereby reducing the number of partitions for a given genomic data. As a result, it takes less time to process and encrypt. For example, applying $PB = 400$ to partition a VCF file of 631,956 chromosomal positions will give 1080 partition blocks whereas $PB = 1000$ will yield 682 blocks. Hence, less time is needed to process and encrypt 682 blocks than 1080 blocks. It can also be observed that CI records the highest runtime across all partition blocks. This observation is a result of genomic data preprocessing at CI i.e. partitioning data into blocks, updating the bloom filter ($\mathcal{BF}$) and encrypting each partition block as shown in Figure 4.8b. Figure 4.8b reveals that the computation cost to partition the genomic data into blocks and update the $\mathcal{BF}$ is nearly invariable for increasing values of PB. However, the encryption time decreases linearly as PB grows which again supports that fact that less time is required to encrypt the partition blocks as PB increases. It is important to note that the runtime on CI, which is less than $19s$, is only a one-time cost and as such does not affect the practical performance and scalability of our scheme.

4.6.2 Genomic Discovery Query Response Time

The query response time (QRT) of our proposed genomic discovery scheme is the time elapsed just before the DU sends a query to when a response is received. Table 4.1

shows the query response time as the average processing time (over 10 runs) incurred by the parties DU, CI, SN, and CS i.e (QRT $= \lambda_{DU} + \lambda_{CI} + \lambda_{SN} + \lambda_{CS}$) where $\lambda_{DU}, \lambda_{CI}, \lambda_{SN}$ and λ_{CS} are the query processing times on parties DU, CI, SN, and CS respectively. As depicted in Table 4.1, we evaluate the response time with respect to increasing the genomic partition block sizes (i.e. PB $= 200, 400, ..., 1000$). Our results suggests that the query response time increases linearly on nodes CI, SN and CS as PB grows larger where the lowest response time is $8569.55ms$ for PB $= 200$ and the highest is $9515.06ms$ for PB $= 1000$. Our observation supports the fact that the number of variants packed into the ciphertext of a partition increases with an increasing PB and hence requires more time to perform homomorphic computation. It is also evident that the query processing time at DU is nearly invariable as PB increases. This is because formulating and encrypting the query and probe bloom filter on DU is independent of PB. For each PB in Table 4.1, the highest query processing runtime is recorded by the cloud servers (SN and CS) while the lowest is incurred by DU and CI i.e. $\lambda_s > \lambda_{CI} > \lambda_{DU}$ where $\lambda_s = \lambda_{SN} + \lambda_{CS}$ is the total cost on cloud servers (SN and CS). For example, for PB $= 400$, $\lambda_s = 8646.22ms, \lambda_{DU} = 20.62ms$ and $\lambda_{CI} = 118.96ms$. Our observation of a higher runtime cost on cloud servers is in line with a secure outsourced computation model [125] where the aim is to securely offload computation from client-side to high-end computation environments such as the cloud.

Table 4.1: Genomic discovery query response time (milliseconds) on parties DU, CI, SN and CS by varying PB

PB	**Parties**				**QRT**
	DU	**CI**	**SN**	**CS**	
200	22.11	64.16	8386.98	96.30	8569.55
400	20.62	118.96	8466.61	179.61	8785.81
600	23.77	178.38	8575.94	289.52	9067.60
800	22.88	234.41	8666.88	366.31	9290.48
1000	21.60	282.56	8755.31	455.59	9515.06

Notes: **PB**: Genomic partition block size; **QRT**: Query response time

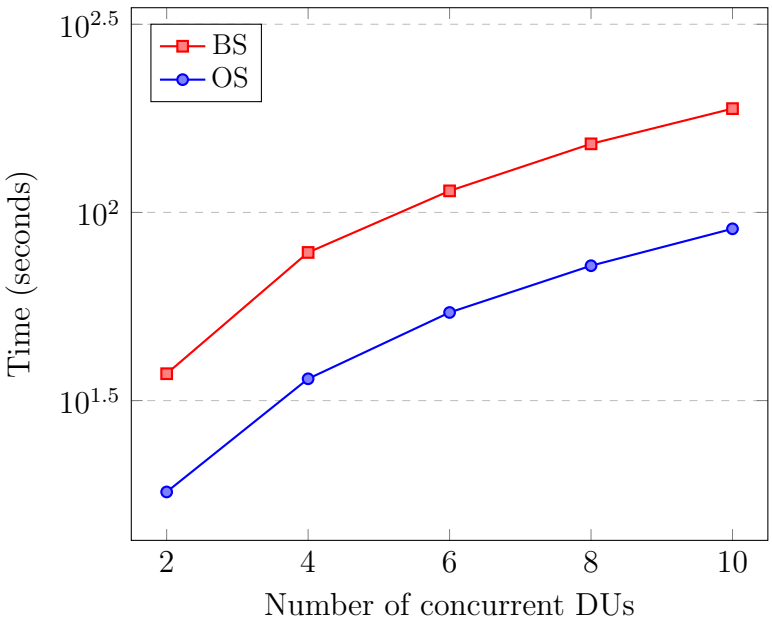


Figure 4.9: Comparing genomic discovery query response time between our proposed optimized scheme (OS) and basic scheme (BS), i.e. an implementation of our scheme without optimization techniques, with a varying number of concurrent genomic data users (DUs).

Optimized query performance

Our proposed scheme optimizes the query response time in two ways. First, the bloom filter proposed in Section 4.4.4 answers the existence of a query variant in SN in $O(k)$ time which is independent of PB and γ (the number of positions in the reference genome). Second, our genomic partitioning technique proposed in Section 4.4.1 reduces the query response time by orders of magnitude. This is because the search space across a DO's entire genomic data is trimmed down from γ to PB. For example from Table 4.1, $PB = 200$ has the lowest query time (i.e. $QRT = 8569.55ms$) because it has a reduced search space of 200 chromosomal positions as opposed to $PB = 1000$ with a search space of 1000 positions which yields the highest $QRT = 9515.06ms$. To further justify this claim, we compare the query response time of our optimized and basic scheme (i.e. an implementation of our scheme without our proposed optimization techniques) with a varying number of concurrent DUs without multi-threading techniques in Figure 4.9. Our results in Figure 4.9 suggest that our proposed optimization techniques reduce the query response time by orders of magnitude. In addition, we compare the query response time of our scheme with

Table 4.2: Query runtime comparison of our scheme **DGD** with current state-of-the-art scheme **GDP** [145]

Scheme	**Query Complexity**	**Query Response Time**
GDP [145]	$O(\upsilon)$	$\approx 74.7s$
DGD	$O(log\frac{\upsilon}{PB} + PB)$	$\approx 20.71s$

Notes: **DGD**: Discover genomic data; **GDP**: Genomic discovery protocol ; υ: number of genetic variants; **PB**: Partition block size. We set PB=2000 and run query over dataset of 28 billion genetic variants as in [145]

the related work [145] as presented in Table 4.2. While [145] takes $\approx 74.7s$ to run a query over 28 billion genetic variants, our technique yields an improved response time of $20.71s$. We exclude the works [143, 144] from our comparison because they are proprietary companies, and their implementations are not public.

Scalability

We also evaluate the scalability of our scheme by setting up multiple concurrent DUs to query the system. We vary the number of concurrent DUs and PBs, and apply multi-threading techniques to process the queries concurrently. We present the response time in Table 4.3 which suggests that for all values of PB, the query response time is nearly invariant to increasing the number of concurrent DUs from 2 to 10. This shows that the query performance of our scheme is not degraded by smaller numbers of concurrent DUs. We would like to acknowledge that in production environments it is natural for applications to process thousands of queries per second. In such cases parallel computation techniques can be applied to process queries in parallel.

Table 4.3: Genomic discovery query response time (milliseconds) by varying the number of concurrent DUs per genome partition block

# of con. DUs	**Genomic partition block sizes**				
	200	**400**	**600**	**800**	**1000**
2	8544.06	9033.40	9735.36	10741.03	11950.14
4	8564.52	9070.71	9820.62	10808.24	12083.91
6	8567.65	9070.65	9834.98	10848.35	12091.43
8	8585.69	9114.97	9876.44	10890.87	12129.07
10	8597.49	9133.62	9909.77	10944.95	12220.02

Notes: #: number; **con.**: concurrent

Our framework currently supports standard precise variant (SPV) queries. However DUs can query for multiple variants in one query request by sending a query object with multiple SPVs. We acknowledge that this might cause excessive transmission cost between parties in the framework. Our framework can be extended in future to pack multiple query variants into one ciphertext.

4.7 Conclusion

In this chapter, we propose a novel secure ecosystem by utilizing blockchain, smart contracts and FHE, which allows a DO to upload their genomic data, and allows DUs to request access to this data. Our scheme also proposes a novel technique for DUs to discover DOs with genetic variants of interest by running secure queries based on FHE, SMC and verification interactions with the BC. We further optimized and reduced the query search space using a novel approach based on genomic partitioning, binary search trees and bloom filters. We implement our framework and demonstrate through extensive experiments on a real-world genomic dataset that our proposed scheme is efficient in terms of computation cost, query response time and scalability while protecting the privacy of DOs. Our framework might result in excessive transmission cost for multiple SPV queries sent by DUs in one request. In future work, we will extend our framework to pack multiple query variants into one ciphertext to reduce the communication cost of multiple SPV queries.

Chapter 5: Blockchain Functionalities for Privacy and Integrity in Genomic Analyses

Blockchain is an emerging distributed ledger technology with the functionalities of decentralization, non-repudiation, transparency, and immutability that is revolutionizing areas in healthcare and finance to ensure the integrity, privacy, and auditability of transactions without a central authority. Recently, the integration of blockchain functionalities for performing genomic and phenotypic analytical task has shown promise in transforming the traditional genomic access control model. The traditional model utilizes middlemen to mediate between genomic and phenotypic data owners, and data consumers. This traditional approach faces challenges of genomic privacy, lack of compensation and access control features. In this chapter, we examine the challenges of the traditional model and investigate state-of-the-art blockchain functionalities to address these problems. First, we explore the most relevant blockchain storage and consent models for genomic and phenotypic data storage, and the encryption techniques used to protect privacy on the blockchain. We then examine and classify state-of-the-art blockchain-based genomic and phenotypic access control and analytics techniques by their application domain. We analyze and compare these techniques to evaluate their security requirements, blockchain functionalities and transaction complexities. Finally, we examine and discuss the limitations and future work on integrating blockchain functionalities to ensure genomic and phenotypic data privacy and analyses.

NOTE: The content of this chapter has been submitted to *ACM Computing Surveys*.

Mohammed Yakubu, A., & Chen, Y. P. P. Blockchain Functionalities for Privacy and Integrity in Genomic Analyses. *ACM Computing Surveys*. (Under Review).

5.1 Introduction

The recent growth of genomic big data as a result of new improvements in DNA sequencing technologies has been an enabler in advancing precision medicine, drug discovery, diagnostic testing and biomedical research. Genomic data goes through a pipeline where it is collected from DNA donors by middlemen, and shared with researchers and clinicians for analysis towards the discovery of new drugs, better understanding of complex diseases etc. This pipeline is the traditional model of acquiring and sharing genomic data which utilizes a conventional approach where middlemen (i.e., third parties such as a direct-to-consumer genomic companies and medical facilities) mediate between genomic data owners and data consumers. The middlemen are either (i) non-profit organizations such as a research organization (e.g., 1000 genome project [3] etc.) to whom individuals willingly donate their genomic data to aid in research, or (ii) for-profit entities including direct-to-consumer (DTC) genetic companies (e.g., 23andMe) who sell genetic services to their customers. DTC's operate a business model where they collate genomic data of their customers and sell them to pharmaceutical and biotechnological companies without compensating their customers [136]. Phenotypic data which are vital in increasing the utility of genomic data are typically used together with genomic datasets in genome-wide association studies (GWAS) to evaluate the correlation between genetic markers and phenotypic traits. Phenotypic data such as disease symptoms, medical history, allergies, radiological images etc., are collected by DTCs together with genomic data via surveys or by healthcare facilities during preventive care and treatment of diseases. The challenges of the aforementioned traditional approach of acquiring genomic and phenotypic are as follows:

1. **Access control:** who has access to the genomic or phenotypic data. Oftentimes, genomic or phenotypic data collectors (i.e. the middlemen) such as DTCs share or sell data access to other third-parties such as pharmaceutical companies without consent from DNA sample donors. This creates a problem where DNA donors lose control and ownership of their data.

2. **Security and privacy:** the security of genomic and phenotypic data, and the privacy of the DNA donors are at risk as they contain sensitive information. For instance, the human genome contains highly sensitive information about an individuals pedigree, his predisposition to diseases etc., and when leaked it might cause genetic discrimination at work or in health insurance [109].

3. **Interoperability:** lack of interoperability amongst genomic and phenotypic data custodians especially amongst heterogeneous healthcare storage facilities. Healthcare facilities that store patients health records and phenotypic data on siloed storage nodes lack the ability to share data efficiently amongst themselves thereby resulting in delayed and inefficient treatments.

4. **Compensation or incentive:** lack of compensation or incentive for DNA donors from genomic data collectors for selling and making profit off their data. In most cases, DTCs do not share proceeds from genomic data sales with the DNA donors. For instance, 23andMe did not compensate their customers for allowing access to their genomic data from their recent $300 million deal with the pharmaceutical giant GlaxoSmithKline (GSK) [153]. This further raising new privacy and data ownership concerns for their consumers.

In addition to these challenges, the traditional model comes with issues of poor genomic and phenotypic data auditability pipelines and lack of data integrity and transparency. Thus, there is a need for genomic and phenotypic data owners to control their data and sell access to it, while protecting their privacy and ensuring the integrity of their data. As a solution to the above challenges, researchers have proposed the use of the blockchain technology and cryptography to address the issues of genomic and phenotypic data sharing, access control, privacy and incentivization challenged by the traditional model. They propose that the blockchain features of immutability, transparency and decentralization would allow genomic and phenotypic data owners to store, own their data and get compensated for sharing data access while maintaining their privacy. The blockchain is a decentralized peer-to-peer network of nodes based on distributed ledger technology (DLT) which stores an immutable and tamper-resistant ledger of transactions between parties in the network. As an emerging technology,

blockchain has many applications in finance, supply chain, healthcare etc., without the need for trust.

In this chapter, we investigate and evaluate the current state-of-the-art techniques based on blockchain features and functionalities to address the challenges of the traditional genomic and phenotypic data sharing model to empower data owners to control their data. Related articles to solve the challenges of the traditional model each present a different contribution with much focus on how healthcare has evolved with the blockchain technology. Current issues on the application of blockchain technology to healthcare have been explored in [154]. This study demonstrates how blockchain can address these healthcare challenges in three areas (i) patients data (i.e. phenotypic data) exchange, (ii) smart contracts for healthcare data tracking, processing and storing on the blockchain, and (iii) healthcare supply chain management to make the tracking of high-value items transparent on the blockchain's shared digital ledger. Another recent survey article with a different focus examines the evolution of healthcare utilizing blockchain technology to address the issues of access control, security, privacy, information sharing and identity management from the perspective of a business ecosystem [155]. This study does not explore the functional technicalities of cryptographic systems and the blochchain network. Other articles review and discuss the challenges faced by the traditional healthcare ecosystem in terms of access control, data provenance, data integrity and interoperability to exchange data between institutions [156, 157, 158]. These studies further examine blockchain techniques such as smart contracts that have been proposed to facilitate data exchange between healthcare institutions while maintaining patients privacy. However, there are very limited studies that investigate blockchain-based techniques for genomic data sharing and access control. A recent study in this direction classified the use cases of blockchain in genomics into distributed computation, data storage and distribution, voting, identity and ownership, and decentralized autonomous organizations [24]. This study further investigates Coinami, a blockchain distributed grid computation framework, which distributes high throughput sequencing (HTS) read mapping task amongst a group of miners. Miners are then rewarded in cryptocurrency for successfully completing an

HTS job.

Note that the scope and focus of these related surveys are different from our main focus, as we investigate and classify a wide range of blockchain features and models developed for genomic and phenotypic data sharing and access control. Specifically, in contrast to the aforementioned related surveys, we investigate and categorize the most relevant storage, consent and compensation models which are vital to realizing the DLT model of acquiring and sharing genomic and phenotypic data. Furthermore, we examine and evaluate current state-of-the-art techniques proposed in literature to integrate blockchain to address the limitations of the traditional model. In summary, the goals of this chapter are as follows:

- We investigate the most relevant storage models employed by blockchain-based techniques to store genomic and phenotypic data and classify them based on their storage architecture into decentralized on-chain storage, centralized off-chain storage and decentralized off-chain storage. In addition, we compare them in terms of their storage and scalability overheads.

- We examine and discuss state-of-the-art compensation models employed in literature to incentivize genomic data owners to contribute their DNA to genomic repositories. We categorize them into cryptocurrencies and tokens, genetic services, dividends and cash.

- We investigate and present the most relevant consent mechanism used by genomic and phenotypic data owners and patients to consent to sharing their data with medical facilities and third-parties. In addition, we classify them by their flexibility to grant and revoke data access into grant only and dynamic consent.

- We discuss and classify functionalities of blockchain for phenotypic data sharing and analytics and group them by their application domain into phenotypic data sharing and access control, phenotypic data collection and processing via medical IoT devices, and machine learning modeling on phenotypic data. We then present a comparative analysis in terms of storage, consent models, blockchain network type, security and blockchain transaction complexities.

- We investigate and categorize current state-of-the-art blockchain-based techniques for sharing and processing genomic data, and classify them by their genomic application domain into secure collection and sharing of genomic data, and querying genomic data access logs on the blockchain. We evaluate and compare these techniques based on their security requirements, blockchain functionalities and query response time.

- Finally, we discuss the challenges, future directions and research gaps in utilizing blockchain features and functionalities to share and process genomic and phenotypic data.

The remainder of this chapter is organized as follows. In Section 5.2, we discuss the preliminaries on blockchain technology. Section 5.3 presents a classification of the most relevant storage models utilized by blockchain-based techniques to store genomic and phenotypic data. We present and discuss the most relevant current compensation models proposed in literature to facilitate genomic data collections and sharing in Section 5.4. A categorization of consent models for allowing DOs to grant or revoke data access in present in Section 5.5. Section 5.6 investigates the cryptographic techniques that have been utilized in literature to protect the privacy of genomic and phenotypic data on the blockchain. Section 5.8 surveys state-of-the-art blockchain-based applications for phenotypic data sharing and analytics, and compares them in terms of blockchain functionalities. Most relevant blockchain-based techniques and ecosystems proposed to share and process genomic data are survey and compared in Section 5.9. Open issues and challenges are present in Section 5.10 and the chapter is concluded in Section 5.11.

5.2 Background on Blockchain

The blockchain is a distributed ledger of transaction records based on decentralized technology. Transactions are validated using digital signatures and packed into blocks which are then chained to the blockchain's ledger using consensus algorithms and cryptographic hash pointers. This ensures the ledger is immutable and tamper-resistant.

Blockchain technology was initially used for cryptocurrency to transfer value (i.e. bitcoin) between network nodes or peers [159]. Following the success story of blockchain in the financial sector (i.e. bitcoin) it is now applied in areas such as healthcare, supply chain, internet of things (IoT), law enforcement etc., to enable transactions between parties without the need of trust. Important features and functionalities of the blockchain that have been utilized to control access, manage, and share genomic and phenotypic data are as follows:

1. **Smart contacts**: Smart contracts are implemented on blockchains to automatically execute agreements between transacting parties without any third-party's involvement. Majority of studies on the application of blockchain to collect and share genomic and phenotypic data have deployed smart contracts to enforce agreements such as consenting to share data, requesting, granting and revoking access to data [160, 161, 140, 162].

2. **Immutable:** Transactions cannot be changed (i.e. modified or deleted) once they have been written to the blockchain's ledger. Adding transactions to the ledger requires that more than 51% of the nodes agree they are valid. This makes it impossible to forge the ledger without consent from majority of the nodes which ensures data integrity.

3. **Decentralized:** The blockchain network is decentralized meaning that it is not controlled by a central authority or a single person. A group of nodes maintain the network and a failure on one node does not compromise the entire network. This makes it resilient to single-point-of-failure attacks to guarantee service availability. In literature, studies have proposed the use of blockchain decentralization to ensure the availability of medical health services for retrieving patients medical images (magnetic resonance imaging (MRI), computed tomography (CT) scans etc.) from an off-chain data storage node [163].

4. **Non-repudiation:** Non-repudiation ensures that the sender of a transaction cannot deny sending the transaction. This is implemented using digital signatures where the sender signs the transaction with his private key and the receiver

verifies the signature with the sender's public key.

5. **Transparency**: The decentralized and immutable ledger of blockchain makes transactions transparent to network nodes. In the case of a private or permissioned blockchain, transactions are only transparent to permissioned nodes. This ensures that data recorded on the blockchain can be audited especially for access log transactions pertaining to requesting for and viewing genomic and phenotypic data by medical and research facilities [145, 164, 165].

5.2.1 Consensus Algorithm

Consensus algorithm is a process by which blockchain nodes reach an agreement on the state of transactions and the distributed ledger without involving any third party or a central authority. Many consensus algorithms have been proposed and applied in distributed ledger technologies. However, in this section we discuss the consensus mechanisms that have been applied to reach agreements on the blockchain's state in the area of genomic and phenotypic data collection, sharing and processing.

1. **Proof-of-work (PoW):** PoW selects a node (i.e. miner) to generate and add the next block to the blockchain ledger. The process of selection involves nodes in the network to solve a complex mathematical puzzle which requires lot of computation power. The first node to solve the puzzle earns the right to add the next block. PoW consensus algorithm has been implemented in networks such as bitcoin.

2. **Proof-of-stake (PoS):** In this mechanism, nodes are selected to validate blocks based on the amount of coins they stake (i.e. their economic stake) in the blockchain network. As opposed to PoW, this approach does not consume lot of energy resources and was proposed as an alternative to PoW.

3. **Practical Byzantine fault tolerance (PBFT):** PBFT provides a practical byzantine state machine replication that can work in the presence of malicious nodes in a blockchain network. In PBFT, consensus is reached on validating blocks using the majority rule. That is nodes in the blockchain network transmit

Table 5.1: Classification of blockchain network architecture types

Blockchain network	Architecture	Participants	SC	Incentive Mechanism	Transaction Transparency
Public blockchain	Decentralized	Anyone can join the network	Slow	Required	Publicly accessible
Private blockchain	Partially decentralized	organizational members	Fast	Not required	Restricted to authorized members within an organization
Consortium blockchain	Partially decentralized	Consortium members	Medium	Not required	Restricted to Consortium members

Notations: **SC**: Scalability

messages among each other to commit a block and the block is only committed to the ledger if it is confirmed by majority of the nodes.

5.2.2 Blockchain Network Architectures

In literature, blockchain networks that have been proposed to control access to genomic and phenotypic data can be classified into three architecture types i.e. Public, private and consortium blockchain architectures [16]. Table 5.1 presents the classification of blockchain network types which are compared in terms of their architecture, the participants that can join the network, scalability, incentive mechanism for mining or adding blocks to the network, and the transparency of transactions data.

1. **Public blockchain:** This is a permissionless blockchain where anyone can join to read and write transactions to the network. It is completely decentralized and any node can participate in the consensus process to validate blocks for a reward. Examples of these types of networks are bitcoin [166] and ethereum [167].

2. **Private blockchain:** A private blockchain is a permissioned network that restricts access of the network to a certain group of participants. This type of blockchain is usually built and maintained by organizations that want to run blockchain services within their organizational setting while restricting access to the public. An example is Hyperledger Fabric [168].

3. **Consortium blockchain:** This blockchain has some features of both a public and private blockchain which is maintained by a group of organizations or com-

panies. Instead of all nodes being able to participate in the consensus process as in the public blockchain, a subset of nodes within the consortium are pre-approved to validate blocks. Consortium blockchains can be built with Quorum [169], Hyperledger Fabric [168] etc.

5.3 Storage Models to Store Genomic and Phenotypic Data on the Blockchain

In the literature on blockchain-based applications for sharing genomic and phenotypic data, the type of storage mechanism has been a concern. This is because (i) genomic and phenotypic data are highly sensitive, (ii) the human genome requires large storage space which can be approximately 200GB depending on the sequencing coverage, number and length of reads, and (iii) the blockchain is constrained in storage as it is expensive and comes with scalability issues. In this section, we categorize the storage techniques applied in literature into the three most relevant storage models i.e., decentralized on-chain storage, centralized off-chain storage and decentralized off-chain storage. We compare these models in terms of their storage cost, scalability, advantages and limitations as presented in Table 5.2.

1. **Decentralized on-chain storage:** This storage model utilizes the blockchain as a database to store patients genomic and phenotypic data. This is depicted in Figure 5.1. The patients data is encrypted or hashed and stored in the data field of a blockchain transaction [173, 174, 175, 162, 170]. Other techniques have defined a customized data structure within a transaction or smart contract to store patients data [176]. The challenges of this storage approach on the blockchain are (i) storage cost e.g., data storage on the ethereum blockchain cost an amount in gas price, and (ii) scalability issues i.e., as transactions and data grows, the less scalable the blockchain becomes.

2. **Centralized off-chain storage:** This storage approach moves patients genomic and phenotypic data storage to an off-chain centralized storage facility such as cloud storage (e.g., AWS, Microsoft azure and Google cloud) or an on-premise secure storage facility while storing pointers or references to the storage

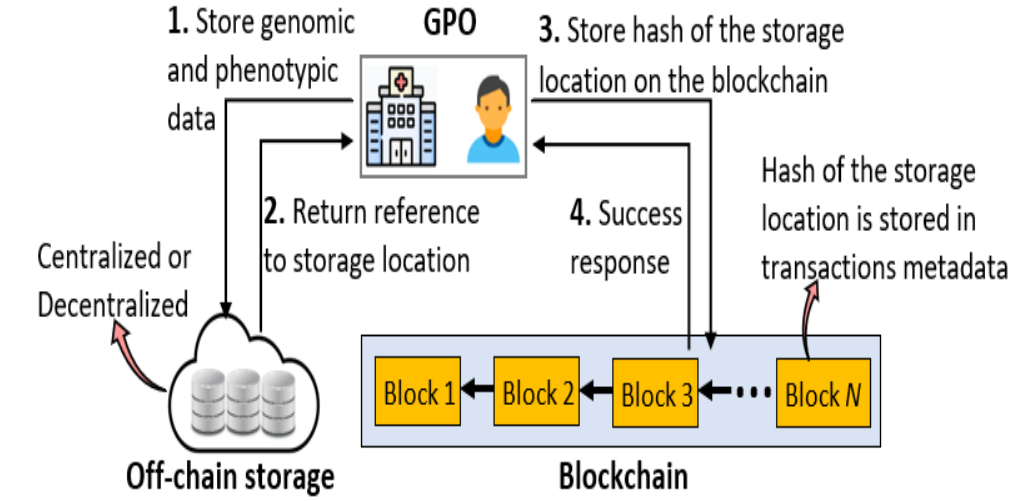
Table 5.2: Comparison of storage models to store genomic and phenotypic data on the blockchain

Storage model	Overheads		Advantages	Limitations
	ST	SC		
DON [170]	H	L	Stores hash of patients phenotypic data, metadata and pointers to storage location on-chain.	High storage cost, scalability issues, non-compliant with GDPR's right to data erasure.
COF [171]	L	M	Stores encrypted patients genomic and phenotypic data on centralized off-chain storage such as the cloud, compliant with GDPR's right to data erasure.	Single point of failure attack, transmission overheads between blockchain and off-chain storage.
DOF [172]	L	H	Stores encrypted patients genomic and phenotypic data on decentralized off-chain storage such IPFS, compliant with GDPR's right to data erasure, addresses single point of failure attack.	Transmission overheads between blockchain and off-chain storage.

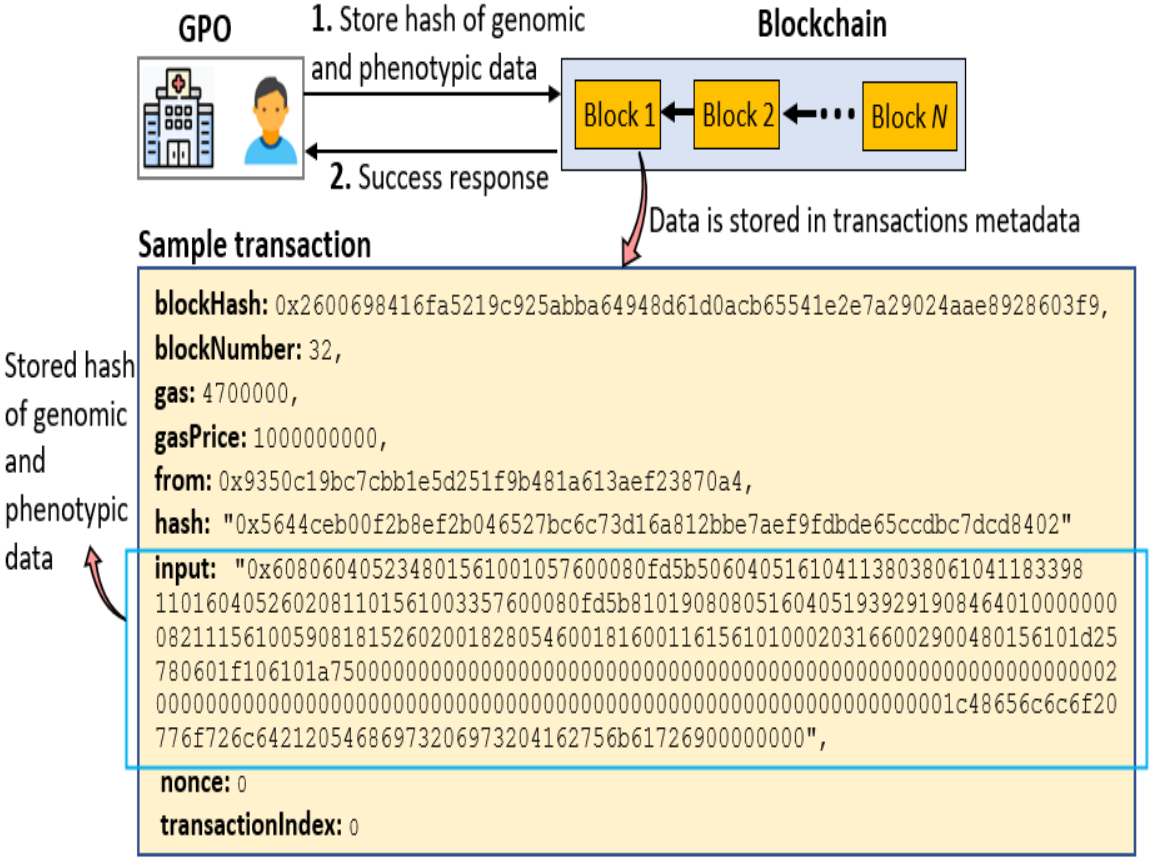
Notations: **COF**: Centralized off-chain storage; **DON**: Decentralized on-chain storage; **DOF**: Decentralized off-chain storage; **GDPR**: General data protection regulation; **IPFS**: Interplanetary file system; **H**: High; **L**: Low; **M**: Medium; **ST**: Storage; **SC**: Scalability

location on the blockchain. Figure 5.1 shows this storage approach. Recent studies have proposed this approach to address the scalability challenges of storing data on-chain [141, 163, 161, 140, 171, 160]. However, off-chain centralized storage still suffers from single point of failure problem i.e., if the storage facility is compromised by an attack, the genomic and phenotypic data becomes unavailable.

3. **Decentralized off-chain storage:** This storage model as shown in Figure 5.1 addresses (i) the scalability limitations of the decentralized on-chain model, and (ii) the single point of failure problem of centralized off-chain storage model by storing genomic and phenotypic data off-chain in decentralized storage nodes [139, 172, 177]. References to storage locations are also stored on the blockchain. Recent studies have utilized this storage model to securely store data on decentralized storage facilities such as InterPlanetary File System (IPFS) [178], Storj [179], Swarm [180], while storing the hash of references to the storage locations on the blockchain.



(a) Centralized or decentralized off-chain storage



(b) Decentralized on-chain storage

Figure 5.1: Storage models to store genomic and phenotypic data on the blockchain. **(a)** Centralized or decentralized off-chain storage: Genomic and phenotypic data are stored on an off-chain storage node while reference to storage location is stored on-chain. **(b)** Decentralized on-chain storage: Genomic and phenotypic data are stored on-chain in transactions metadata. ***Notations:*** **GPO**: Genomic and phenotypic data owner

5.4 Compensation Models to Facilitate Genomic Data Sharing

Compensation models to facilitate genomic data sharing is a system outlined to motivate individuals with a financial or non-financial reward in exchange for their genomic data. The human genome is increasing in value, and as a result several compensation models have been proposed in literature outlining how and why DTCs should incentivize individuals who contribute their genomic data to their repositories [181]. This approach will not only enable DNA data contributors to share in some of the profits, but also encourage and empower them to contribute genomic data to drive research. In this section, we discuss the compensation models proposed in literature to encourage the sharing of genomic data and group them into the four most relevant reward types i.e., (i) cryptocurrency and tokens, (ii) genetic services, (iii) dividends based on DNA and health data types, and (iv) one-time cash payment. We illustrate categorization of these compensation models in Figure 5.2.

5.4.1 Cryptocurrencies and Tokens in Exchange for Genomic Data

A cryptocurrency is a digital asset or currency created and secured by cryptography on a blockchain network to store value and facilitate transactions for the payments of goods and services. In literature, genomic data sharing ecosystems built on top the blockchain network have proposed cryptocurrencies as a means to compensate genomic data owners as opposed to the use of fiat currencies in the traditional model [24, 25]. These ecosystems enforce a data access or transfer agreement between the genomic data owner (DO) and the genomic data consumer (DC) which guarantees that the DC issues a one-time payment to the DO upon receiving data. These agreements are enforce via smart contracts which are immutable meaning that terms and conditions of the agreement cannot be changed once deployed on the blockchain network. Recent blockchain ecosystems for sharing genomic data that have implemented this compensation model are Nebula Genomics [144], Zenome [143] and Shivom [182].

Nebula Genomics, Zenome and Shivom have issued their own platform dependent tokens as opposed to using the native cryptocurrencies of the blockchain platforms they were built on such as bitcoin (BTC) and ether (ETH). Whereas Nebula Genomics have issued Nebula tokens as the currency for compensating DOs in their ecosystem, Zenome and Shivom have implemented Zenome DNA (ZNA) and OmiX token respectively.

5.4.2 Genetic Services for Compensating Genomic Data Owners

Genetic services such as DNA sequencing, genetic testing for ancestry information, disease predisposition etc., are offered on the traditional model by DTCs for a fee payable in fiat currency. Individuals send their DNA sample (e.g. saliva) to DTCs and pay for DNA sequencing followed by additional services they have ordered for such as ancestry information at an extra cost. This creates a scenario where DTCs acquire multiple revenue streams from their customers with no form of compensation [25] i.e., DTCs acquire revenue from (i) DNA sequencing and genetic testing services, and (ii) monetizing customers genetic data by selling them to pharmaceutical companies. A recent study has demonstrated that the blockchain technology not only provides a way to democratize genomic data sharing, but also allows fees charged for genetic services to be waived or subsidized as a means to compensate and encourage individuals to share their data [183]. This compensation approach has been adopted by Nebula Genomics [144] where researchers connect with individuals with certain traits of interest, and offer to subsidize the cost of sequencing their DNA. Another blockchain genomic ecosystem to compensate with genetic services is Shivom. Shivom's ecosystem invokes a smart contract to allow DCs to partially or fully (i) subsidize genomic testing, or (ii) reimburse individuals who have already paid for the test.

5.4.3 Dividends Based on DNA and Health Data Types

The net revenue generated from monetizing genomic data repositories are distributed as dividends amongst shareholders of genomic data collection companies and DTCs.

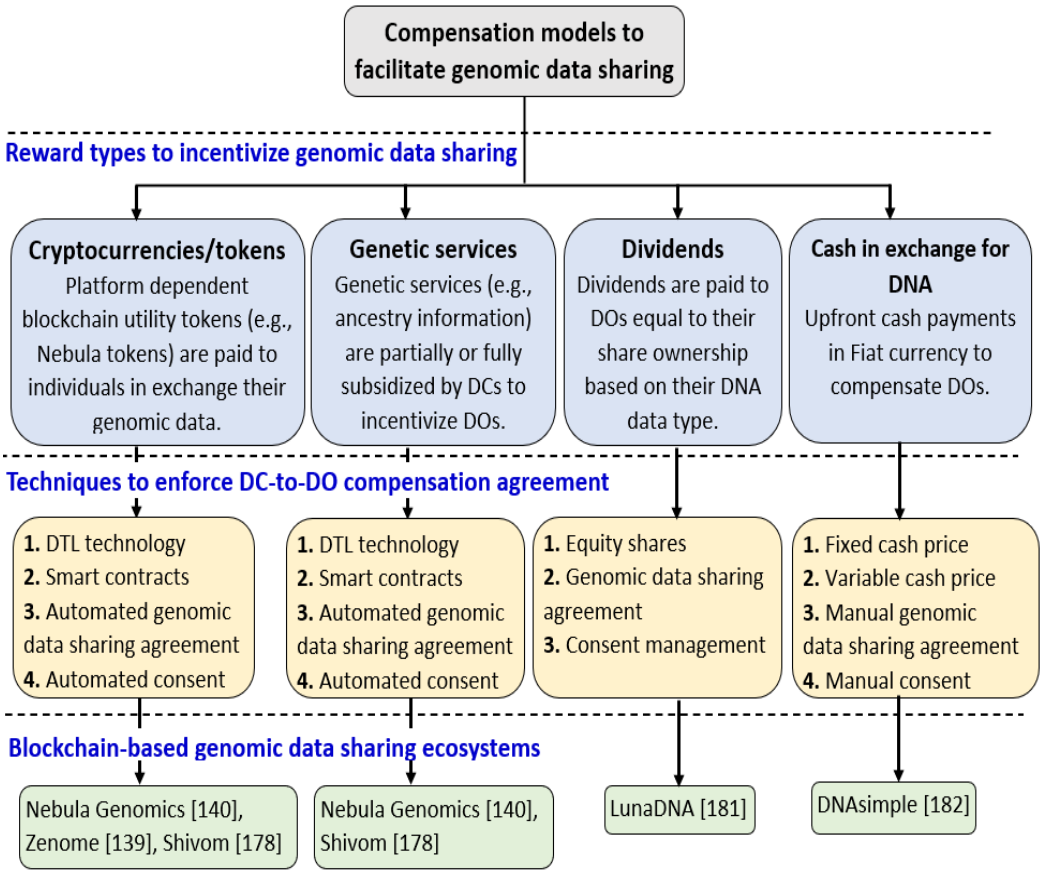


Figure 5.2: Categorization of compensation models to facilitate the sharing of genomic data classified based on the means that DOs are incentivized into cryptocurrencies and tokens, genetic services, dividends, and cash in exchange for DNA. ***Notations:*** **DO**: Genomic data owner; **DC**: Genomic data consumer; **DLT**: Distributed ledger technology

The distribution of dividends do not include the individuals who contributed their genomic data. This problem is inherent to the traditional model where DOs do not share in some of the profit. A recent study sought to address this problem by proposing a new model to compensate DOs with dividends from proceeds of genomic data [184]. The LunaDNA ecosystem has been built around this notion of compensating DOs [185]. Individuals who contribute their data to LunaDNA's repository earn shares of the company based on their data types such DNA Exome, DNA Whole Genome, DNA Tumor Targeted Genes etc. For example, an individual who contributes DNA Exome earns 150 shares, while DNA Whole Genome earns 300 shares. After researchers and DCs have paid to access genomic data, LunaDNA then pays dividends to DOs commensurate with their share ownership as shareholders [184].

5.4.4 Cash in Exchange for DNA

In literature, cash upfront payments have been proposed as a means to compensate individuals to share their DNA for research studies [184]. The compensation model for DNAsimple ecosystem is based on this cash-for-DNA approach [186]. This ecosystem match research studies to individuals of interest based on their background. Individuals are compensated with an upfront cash payment each time they qualify and participate in a study with their DNA. Two schemes of cash compensation have been discussed in literature [184]: (i) fixed cash compensation: this is the most basic scheme where a fixed cash price is paid to an individual upon DNA sample collection, and (ii) variable compensation: this compensation scheme varies the cash price depending on the genomic research study and the scarcity of DNA sample as assessed by the data custodian.

5.5 Consent Models to Grant/Revoke Access to Genomic and Phenotypic Data Repositories

Consenting to share genomic or phenotypic data in literature is a means of validating that permission to view, use, and share data with a specific or group of data users has been granted by the data owner. Consent models are a set of permissions and access policy statements on how to use data which are mostly enforced by immutable blockchain smart contracts. This is relevant for developing blockchain-based genomic and phenotypic data access control frameworks as it provides the data owner with a feature to define how he wants his data to be used [187]. We classify the most relevant consent models applied in literature to request access to genomic or phenotypic data into two groups i.e., grant only and dynamic consent.

1. **Grant only:** In this model patients can only grant permission to clinicians and third parties to use their data without the ability to revoke access [175, 139, 163, 162, 188, 170]. This is applied in scenarios where data owners grant a one-time access to their data, and data users download or transfer data to their private servers for processing. This model is limited by the lack of flexibility for patients

or data owners to fully control their data.

2. **Dynamic consent:** Dynamic consenting allows patients and data owners to grant or revoke access to their genomic or phenotypic data anytime [172, 177]. This addresses the limitation of the "grant only" consent model which only allows one-time access approval from the data owner. In literature, smart contracts on the blockchain have been employed to implement dynamic consenting by updating smart contracts state variables to grant or revoke genomic or phenotypic data access at any time based on time-controlled mechanisms [174], access tokens [141] and encryption techniques such as proxy re-encryption [142].

5.6 Cryptographic Techniques to Secure Genomic and Phenotypic Data on Blockchain Networks

In this section, we provide a detailed discussion on the cryptographic techniques that have been leveraged to enhance the security and privacy of existing blockchain systems to store and process genomic and phenotypic data.

5.6.1 Digital Signatures

A digital signature is a cryptographic technique used to verify the integrity and authenticity of a message [189]. These signatures are fundamental building blocks in blockchains which enables the sender of a transaction to prove to other nodes in the network that the request is authentic. A digital signature scheme consist of a triple of probabilistic polynomial time algorithms, $(\mathcal{G}, \mathcal{S}, \mathcal{V})$, where $\mathcal{G}$ is the key generation algorithm used to generate a private and public key, $\mathcal{S}$ is the signing algorithm which uses a given private key to sign a message input, and $\mathcal{V}$ is verification algorithm which validates a messages signature. In the context of genomic data sharing and processing, digital signatures allow an individual to share his genomic data on a blockchain ecosystem while proving to every node that he initiated the transaction and it has not been tampered with. This has been demonstrated in a recent study where researchers proposed the use of digital signatures for integrity checking in the context of genomic

data sharing and consenting [190]. In this study, an individual utilizes his digital signature to prove to a service provider that his shared genomic data is authentic.

5.6.2 Homomorphic Encryption

Homomorphic encryption (HE) techniques enables certain operations (addition or multiplication) to be carried out directly on ciphertext [60]. This allows the storage and computation on confidential genomic data to be outsourced to untrusted environments in encrypted form. HE has been applied to secure genomic data off-chain while allowing researchers to run bioinformatics analysis pipelines such as GWAS pipeline on encrypted genetic variants. In addition to providing secure computation, HE enables a storage model where genomic data does not move, but instead the computational pipeline comes to the data [25]. This avoids the high communication overheads and longer times to transit large volumes of genomic data between parties in the blockchain ecosystem. Limitations of using HE to secure genomic data on blockchains are scalability issues, high computation and communication cost [25, 60].

5.6.3 Differential Privacy

Differential privacy (DP) uses perturbing techniques to add noise sampled from a probability distribution to data in such a way that an adversary cannot re-identify or recover the original data [56]. DP techniques often balance between data privacy, accuracy and utility of the data. In literature, DP has been applied to protect the privacy of query outputs on genomic databases and aggregated statistical result such as minor allele frequencies (MAFs) from genomic datasets managed by blockchains [191, 145]. A randomized algorithm A is ε-differentially private over two databases D_1 and D_2 which differ in at most one record if below is true:

$$Pr[A(D_1) \in \theta] \leq e^{\varepsilon} \times Pr[A(D_2) \in \theta] \tag{5.1}$$

where θ is the possible set of query outputs on the databases, ε is the privacy budget which is a measure of the privacy leakage in differential privacy. A smaller ε

provides higher privacy but lower accuracy. In applications where blockchain access control mechanism is used to manage genomic and phenotypic databases, preserving the privacy of individual records are important. A recent study utilized DP to protect the privacy of individuals whose DNA variants and clinical data are queried from a genomic database controlled by a blockchain [145]. This study obfuscated the query results with noise so that an adversary cannot reconstruct sensitive attributes of individuals.

5.6.4 Secure Multi-party Computation

Secure multi-party computation (SMC) is a privacy-preserving computation technique which aims at enabling multiple parties $P_1, P_2, ..., P_n$ to evaluate a function $g(.)$ on their private data $D_1, D_2, ..., D_n$ [192]. At the end of the protocol each party learns nothing but the computation result $g(D_1, D_2, ..., D_n)$. Implementations of SMC are usually based on (i) garbled circuits where $g(.)$ is expressed as a boolean circuit which is executed with oblivious transfer, or (ii) secret sharing where a secret is randomly shared amongst the SMC parties. SMC has a wide range of blockchain-based applications in securing genomic and phenotypic data. For instance, a recent study employs SMC to privately share features extracted from medical images on a blockchain network. Another study on blockchain-based genomic data sharing utilizes SMC to distribute trust amongst several computing servers [145]. This trust distribution enables the computing servers to build a collective public key for encryption and decryption process where each server sequentially performs a partial decryption with its secret key.

5.6.5 Intel Software Guard Extensions

Intel software guard extensions (SX) is an extension of intel's core microprocessors which allows the isolation of certain application code and data in secure memory regions called an enclave [193]. It has been demonstrated in several studies that SX is more efficient to compute on genomic data than HE and SMC. Furthermore, a hybrid solution based on SX and HE has been applied to address the limitations of HE to

speed up computation of genomic data on the blockchain network. For instance, the nebula network [194] combines homomorphic encryption with SX to improve computation time on genomic data. Their hybrid approach decomposes bioinformatics computations (e.g., testing the significance of genomic variants) to perform addition operations on genomic data encrypted with homomorphic encryption and further arbitrary computations are carried out inside the SX enclave. Despite the low computation and communication cost of SX, it is restricted in memory and prone to security vulnerabilities such as cache attacks, page-table based attacks etc. [195].

In addition to the above cryptographic primitives, encryption techniques including symmetric key encryption (e.g., The Advanced Encryption Standard (AES) algorithm etc.) and public-key encryption (e.g., Rivest, Shamir, and Adelman encryption (RSA), ElGamal encryption system etc.) have also been used to protect the privacy in block-based access control frameworks for sharing genomic and phenotypic data [175, 139, 142, 163, 161, 140, 145].

5.7 Blockchain Protocol for Genomic and Phenotypic Data Access Control

In this section, we present a general protocol for the distributed ledger technology (DLT) approach of sharing and controlling access to genomic and phenotypic data. This protocol has been adopted by several studies [172, 171, 170, 177, 161, 139, 145] that implement the DLT model using blockchain functionalities. We categorize the relevant entities of the general protocol as follows:

1. Genomic and phenotypic data owner (GPO): The GPO is the party who owns the data such as a patient or a DNA donor. It could also be a data custodian such as a genomic data repository, medical facility etc.

2. Genomic and phenotypic data consumers (GPC): The GPCs are third-parties who use the data. Examples of GPCs could be medical facilities such as hospitals, research organizations, pharmaceutical companies etc.

3. Blockchain network (BC): The BC is a decentralized peer-to-peer network that

immutability stores data, metadata and transactions pertaining to managing and controlling the access of GPO's data.

4. Off-chain storage (OS): OS has been employed by frameworks that implement either the centralized off-chain storage or decentralized off-chain storage model to store encrypted data. For the sake of brevity, in this section we use "data" to mean "genomic and phenotypic data"

We classify the general blockchain based approach to share genomic and phenotypic data into four steps as follows:

Step 1. **Initialization**: Before using the secure system, each party requires (i) a pair of public key pk and private key sk to sign and verify blockchain transactions, and (ii) a blockchain address which is tied to their blockchain account to send transactions. Depending on the blockchain types, the key pair and blockchain address are generated for a party after registering on the blockchain network.

Step 2. **Encryption of data**: The GPO encrypts his data i.e., $\mathcal{C} = Enc(\mathcal{D})$, where $\mathcal{C}$ is the ciphertext of $\mathcal{D}$ and $Enc(.)$ is the encryption technique. The encryption technique $Enc(.)$ used depends on the goals and use case of the application. For instance, to achieve computation on the encrypted data a homomorphic encryption technique is used, and to distribute data amongst multiple parties an SMC based technique will be more suitable.

Step 3. **Upload data to the blockchain**: The GPO then uploads the ciphertext $\mathcal{C}$ to the blockchain. Here there are three storage models as presented in Section 5.3 i.e., decentralized on-chain storage, centralized off-chain storage and decentralized off-chain storage.

1. *Decentralized on-chain storage:* For decentralized on-chain storage, GPO computes the hash of $\mathcal{C}$ i.e., $H(\mathcal{C})$, and stores $H(\mathcal{C})$ in the metadata field of a blockchain transaction as shown in Figure 5.1. Recall that this storage technique results in blockchain scalability issues as the number of transactions λ and the data n grows across blockchain ledgers i.e., $O(n\lambda)$.

2. *Centralized off-chain storage or decentralized off-chain storage:* Here GPO stores

$H(\mathcal{C})$ on a centralized server or a decentralized storage facility such as InterPlanetary File System (IPFS). The hash of the storage location $H(\mathcal{L})$ is returned to GPO. GPO then writes $H(\mathcal{L})$ to the blockchain. Recall that this approach shifts data storage from the blockchain to improve scalability.

Step 4. **GPC request data access**: GPC sends a request to the blockchain to request data access from a GPO. GPO receives the request and either grants or denies access to GPC. If GPC is granted access, GPC is provided with the necessary keys and permission rights to view or access the data. Here two general consent models (discussed in Section 5.5) can be implemented (1) the grant-only consent model: the GPO after granting access cannot revoke it, and (2) the dynamic consent model: the GPC can grant and revoke access anytime.

It is important to note that the steps presented above can be automated and seamlessly integrated using smart contracts. For instance, the schemes of [171, 177, 145] have developed smart contact based access control techniques to grant data access to the GPC.

5.8 Functionalities of Blockchain Applications for Phenotypic Data Sharing and Analytics

Patients share their phenotypic data (electronic medical records -diagnoses, medications, treatment plans, immunization dates, allergies, radiology images etc.) with physicians to identify and treat health conditions. The sensitivity of these data requires that health institutions secure and maintain the privacy of patients when data is collected, stored and shared with other health facilities within the healthcare ecosystem. However, this is hindered by the challenges of the traditional phenotypic data sharing model such as security, access control, integrity, provenance and interoperability amongst heterogeneous data sources as shown in Figure 5.3. Recently, researchers have proposed techniques based on decentralized blockchain functionalities (tamper proof, immutable, transparent etc.) to address these challenges. In this section, we discuss and investigate current state-of-the-art techniques that have been proposed to leverage blockchain to address the challenges of the traditional phenotypic data shar-

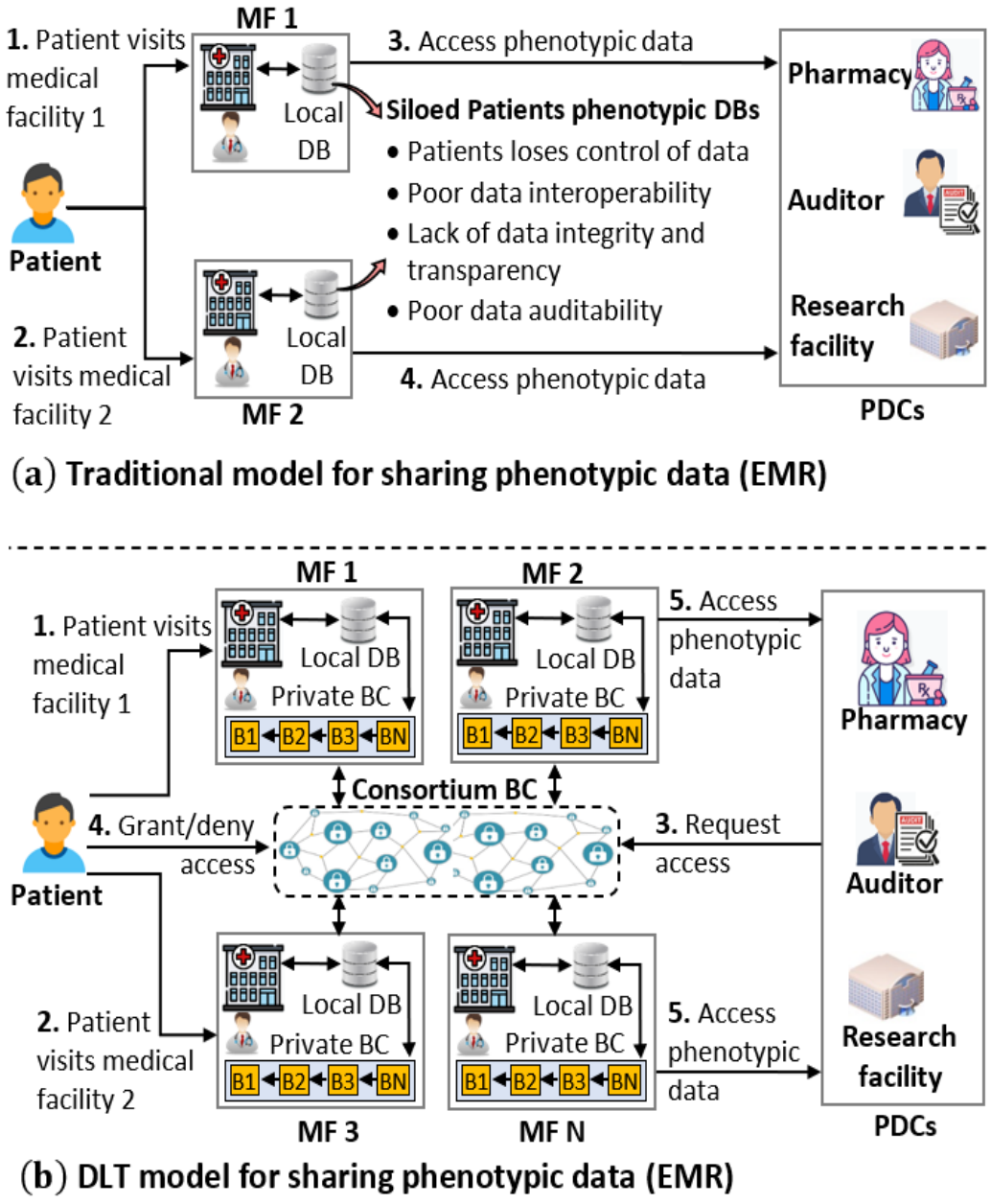


Figure 5.3: Traditional vs. DLT models for sharing phenotypic data. (**a**) Traditional model for sharing phenotypic data where patients phenotypic data are siloed across medical facilities, and creating challenges such as losing access to data, data interoperability, integrity and transparency issues. (**b**) DLT model for sharing phenotypic data where the blockchain is used to give patients control over their phenotypic data within a consortium of medical facilities, and thereby addressing the challenges of the traditional model. Each medical facility has a private blockchain for controlling data access and integrity. ***Notations:*** **BC**: Blockchain; **B1**: Block 1; **B2**: Block 2; **B3**: Block 3; **BN**: Block N; **DLT**: Distributed ledger technology; **DB**: Database; **EMR**: Electronic medical record; **PDCs**: Phenotypic data consumers.

ing model. We classify them by their application domain: phenotypic data sharing and access control, phenotypic data collection and processing via medical IoT devices, and machine learning modeling on phenotypic data. We then compare and evaluate them

in terms of blockchain storage models, security requirements, architecture frameworks and blockchain transaction complexities.

5.8.1 Phenotypic Data Sharing and Access Control

Empowering patients with the ability to control their phenotypic data by granting and revoking access control rights to clinicians and third-parties would allow them to manage their privacy. However, in the traditional healthcare data sharing model, patients phenotypic data are controlled and managed by health facilities who create and define data access control policies. This creates a security and privacy problem as health facilities and other establishments that store patients data are semi-honest and can share data without patients consent. Blockchain features such as smart contracts, non-repudiation and tamper-proof have been leveraged to propose solutions to allow patients to manage their phenotypic data and control who can access or view them. This was demonstrated in a recent study where researchers logged data storage, search and retrieval transactions in the blockchain to provide an immutable evidence of phenotypic data usage and ownership [139]. This study utilizes attribute-based encryption to assign access rights to phenotypic data ensuring that only authorized parties can access patient data. Another study proposes the use of two blockchains (i.e., private and consortium blockchain) to facilitate phenotypic data sharing between health facilities to improve patients diagnosis [174]. The private blockchain stores patients phenotypic data and personal health information, while the consortium blockchain stores searchable indexes of patients phenotypic data. This allows patients to generate searchable trapdoors to enable authorized doctors to access their data.

Blockchain smart contracts makes it easy to program and automate the behaviors of patients and clinicians when consenting to data access and sharing without the need for trust. Phenotypic data consent policies/regulations are programmed in smart contracts which automatically trigger in response to an access request from a clinician or an approval from patients. This has recently been demonstrated by researchers where a dynamic consent model has been developed using smart contracts to standardize pa-

tients consent for sharing phenotypic data [140]. This approach combines two consent models: (i) data use ontology (DUO) to model patients consent over their phenotypic data, and (ii) automatable discovery and access matrix (ADA-M) to model queries from clinicians based on patients phenotypic data use cases and categories.

Patients phenotypic data collected by health institutions during the course of treatments are typically siloed in the institutions data stores. This creates a barrier for different health facilities to interoperate and exchange patients data resulting in delayed and inefficient treatments. Well recognized healthcare data interoperability protocols such as the fast healthcare interoperability resources (FHIR) standard [196] defines how different health facilities can securely share clinical data regardless of their storage formats. Recent studies have developed blockchain-based frameworks to conform to FHIR standard to overcome the barrier of sharing phenotypic data across different health facilities while enabling patients to control access to their data [141, 142]. While [141] develops a token-based access mechanism using smart contracts to comply with FHIR standard to empower patients to grant and revoke access to their data, [142] adopts proxy re-encryption (PRE) via smart contracts to grant and revoke data sharing rights for patients data in accordance with FHIR standard and HIPAA security rules.

Personalized data segmentation enables patients to choose health records they would like to share or keep confidential from clinicians. For example, a patient might grant access of his cardiology data with his clinician while keeping history of his substance abuse confidential for reasons of privacy. A recent study demonstrated the use of the blockchain technology to log permission records to allow patients to segment which category to their phenotypic data to share or grant access [161]. This work also conforms to FHIR standard and allows clinicians to check phenotypic data provenance. However, it is limited by blockchain scalability constraints as the number of transactions (e.g. permission granting) generated might exceed the total number of transactions the blockchain can process per second. It is expensive and inefficient to store data on the blockchain as the size of the entire blockchain grows proportionally to the number of transactional data stored in a block. The efficiency issue worsens

for the storage of image data types such as radiological images (magnetic resonance imaging (MRI), computed tomography (CT) scans etc.). A recent study addressed this blockchain challenge by proposing a solution to enable patients of radiological studies to grant access of their radiological images to authorized parties [163]. The radiological images are stored on a central off-chain data store, while pointers to URL endpoints of the storage location are stored on-chain thereby reducing the storage burden on the blockchain and improving efficiency.

Table 5.3: Blockchain functionality comparison of techniques for phenotypic data sharing and analytics

Category	**Scheme**	**Stor. Model**	**Cons. Model**	**Crypt. Techn.**	**BC Arch.**	**BC Type**	**CA**	**SC**	**Data Priv.**	**TSC**
Phenotypic data sharing and access control	Ichikawa et al., 2017 [173]	DON	N/A	N/A	PB	HLF	PBFT	✓	●	$O(np)$
	Zhang et al., 2018 [174]	DON	Dyn.	PEKS	UB, CB	ETH	PConf	✓	●	$O(nk)$UB+ $O(n\phi)$CB
	Peng et al., 2018 [141]	COF	Dyn.	PKC	N/A	ETH	N/A	✓	●	$O(n(h + \rho))$
	Kleinaki et al, 2018[160]	COF	N/A	PKC	UB	ETH	N/A	✓	●	N/A
	Wang et al., 2018 [162]	DON	grant only	N/A	CB	N/A	DPoS	✓	N/A	$O(nh)$
	Li et al., 2019 [175]	DON	grant only	AE (RSA)	PB	HLF	PBFT	✓	●	$O(ns)$
	Hylock et al., 2019 [142]	hybrid	Dyn.	PRE, CH, AES	PB	HLF	PBFT	✓	●	$O(nk)$
	Patel et al., 2019 [163]	COF	grant only	PEKS	PB	HLF	PoS	N/A	N/A	$O(n\rho)$
	Zhuang et al., 2020[161]	COF	Dyn.	N/A	PB	ETH	N/A	✓	●	$O(n(m + h))$
	Jaiman et al., 2020 [140]	COF	Dyn.	N/A	N/A	ETH	N/A	✓	○	$O(n\rho)$
	Sun et al., 2020 [139]	DOF	grant only	ABE	N/A	N/A	N/A	×	●	$O(n(h + \delta))$
	Cao et al., 2020 [197]	off-chain	N/A	AES	UB	N/A	N/A	×	●	$O(nq)$
Phenotypic data collection and processing via medical IoT devices	Zhang et al., 2016 [188]	off-chain	grant only	ECC	N/A	N/A	N/A	×	●	N/A
	Li et al., 2019 [171]	COF	Dyn.	AES	CB	ETH	N/A	✓	●	$O(n\varphi)$
	Xu et al., 2019 [172]	DOF	Dyn.	RSA, AES	UB, CB	N/A	PoW, PBFT	×	●	$O(nq)$UB+ $O(nd)$CB
	Tomaz et al., 2020 [177]	DOF	Dyn.	ABE, ECC, NIZKP	N/A	ETH	N/A	✓	●	$O(n(m + \delta))$

Continued on next page

Table 5.3 Blockchain functionality comparison of techniques for phenotypic data sharing and analytics - Continued

	Yazdinejad et al., 2020 [170]	DON	grant only	SKE	UB	N/A	PoW	N/A	●	$O(nk)$
	Shen et al., 2019 [176]	hybrid	Dyn.	SMC, PKC	N/A	ETH	N/A	✓	●	$O(n(v + t))$
Machine learning modeling on phenotypic data	Kuo et al., 2019 [198]	COF	N/A	N/A	PB	MLC	N/A	×	○	$O(\beta^2)$
	Kuo et al., 2020 [191]	COF	N/A	N/A	PB	MLC	N/A	×	○	$O(\beta^2+\psi)$
	Kuo et al., 2020 [199]	COF	N/A	N/A	PB	MLC	N/A	×	○	N/A

Notations: **Arch.**: Architecture; **AES**: Advanced encryption standard; **AE**: Asymmetric encryption; **ABE**: Attribute-based encryption; **BC**: Blockchain; **CH**: Chameleon Hashing; **CB**: Consortium blockchain; **Cons.**: Consent; **CA**: Consensus Algorithm; **Crypt.**: Cryptographic; **COF**: Centralized off-chain storage; **DOF**: Decentralized off-chain storage; **DON**: Decentralized on-chain storage; **DI**: Dividends; **Dyn.**: Dynamic consent; **ETH**: Ethereum; **EXO**: Exonum; **ECC**: Elliptic curve cryptography; **ECDSA**: Elliptic Curve Digital Signature Algorithm; **HLF**: Hyperledger Fabric; **MLC**: Multichain; **NIZKP**: Non-Interactive Zero-Knowledge Proof; **Priv.**: Privacy; **PoW**: Proof-of-work; **DPoS**: Delegated-proof-of-stake; **PConf**: Proof-of-conformance; **PoS**: Proof-of-stake; **PBFT**: Practical Byzantine fault tolerance; **PB**: Private blockchain; **UB**: Public blockchain; tolerance; **PRE**: Proxy re-encryption; **PKC**: Public key cryptography; **PEKS**: Public encryption with keyword search; **RSA**: Rivest, Shamir, and Adelman encryption; **SC**: Smart contract; **Stor.**: Storage; **SMC**: Secure multi-party computation; **SKE**: Symmetric key encryption; **Techn.**: Technique; **TSC**: Transaction storage complexity; **N/A**: Not available or applicable; $\boldsymbol{n}$: # of patients; $\boldsymbol{p}$: size of patients phenotypic data; $\boldsymbol{k}$: size of encrypted patients phenotypic data; $\boldsymbol{\phi}$: size of secure indexes of patients phenotypic data; $\boldsymbol{h}$: size of hashed patients phenotypic data; $\boldsymbol{q}$: size of hashed encrypted patients phenotypic data; $\boldsymbol{\rho}$: size of reference to storage location of patients data; $\boldsymbol{\varphi}$: size of encrypted reference to storage location of patients data; $\boldsymbol{\beta}$: number of model covariates; $\boldsymbol{\psi}$: number of the level of the hierarchy; $\boldsymbol{s}$: size of patients prescription records; $\boldsymbol{\delta}$: size of hashed storage address from IPFS; $\boldsymbol{m}$: size of metadata from patients data; $\boldsymbol{d}$: size of hashed encrypted doctors diagnosis; $\boldsymbol{v}$: size of encrypted medical image feature vectors; $\boldsymbol{t}$: size of encrypted patients diagnostic data;

5.8.2 Phenotypic Data Collection and Processing Via Medical IoT Devices

During the collection of phenotypic data (e.g., electrocardiogram (ECG), weight, temperature etc.) from IoT devices such as body sensors, nodes that pre-process these

data prior to sharing them with healthcare professionals are challenged with security vulnerabilities where a malicious application might compromise the patient's data. Recently, researchers have leveraged the transparent feature of blockchain together with cryptographic protocols to develop authentication schemes where the identity of IoT nodes that pre-process patients phenotypic data are authenticated to mitigate security vulnerabilities to protect patients privacy [188, 170]. The technique in [188] proposes a blockchain-based solution to enable patients within a pervasive social network (PSN) healthcare ecosystem to share their data collected by medical sensors for remote medical care. Addresses of medical sensors are recorded in a blockchain transaction to facilitate authentication and phenotypic data access across the PSN network. The work [170] develops a decentralized patient authentication scheme where phenotypic data collected from an IoT device is symmetric key encrypted prior to storage on the blockchain. This allows patients to migrate to affiliated hospitals without the need for re-authentication thereby leading to less delay when sharing phenotypic data across multiple hospitals within the network. Another recent study in this direction applies Non-Interactive Zero-Knowledge Proof to authenticate patients data collected from mHealth devices [177]. In this work a fine-grained access control over patients data is achieved based on the blockchain and attribute-based encryption (ABE).

In smart healthcare delivery, it is essential to protect doctors diagnosis as much as patients data are protected to ensure data integrity in order to avoid medical disputes. A recent study propose the use of two blockchains to ensure patients data and doctor's diagnoses integrity [172]. This work developed (i) a userchain which is a public blockchain to record, read and send transactions pertaining to phenotypic data, and (ii) a docchain which is a consortium blockchain to publish doctor's diagnoses.

Majority of blockchain-based solutions on processing data collected from IoT devices utilize blockchain for data management, authentication of IoT devices and access control. A recent study, however, applied blockchain techniques for secure image retrieval over encrypted patients medical images collected via IoT devices [176]. In this work, feature vectors (edge histogram descriptors) of medical images collected from multiple data owners are extracted, encrypted and stored on the blockchain. Smart

contracts are then deployed to coordinate an image retrieval service to process queries from data users over the encrypted image feature vectors for a similarity match.

5.8.3 Machine Learning Modeling on Phenotypic Data

Patients health records across multiple institutions provides the benefit of creating more generalizable predictive machine learning models on healthcare data to support and accelerate clinical, and biomedical research. Central servers are used to manage the training process across institutions, and to disseminate the trained models instead of the actual sensitive data to protect patients privacy. However, the use of central servers are challenged by (i) unfair allocation of modeling task amongst participating institutions, and (ii) the ability to tamper with the model parameters and source institution of the models on the central server by an attacker.

Recent studies have proposed the use of blockchain to mitigate these challenges. This work applied the decentralized feature of blockchain to allow institutions to exchange machine learning models without the need of a central server. Once models were recorded in transactions metadata on the blockchain, they cannot be tampered with and the source institutions can easily be verified [198, 191]. While [198] focuses on using the blockchain to share computational loads fairly amongst participating institutional sites, [191] develops hierarchical consensus learning algorithm with multiple levels for predictive learning. Another study on blockchain-based machine learning modeling on cross-institutional patients data performs online logistic regression on patients phenotypic data [199]. In this study, participating institutions share partially trained models amongst themselves via the blockchain distributed ledger.

These aforementioned studies protect the privacy of patients by sharing only aggregated machine learning models on-chain. However, there are still some security vulnerabilities where the model may reveal information specific to its source institution leading to inference attacks on patients data. A solution to these security vulnerabilities is to encrypt the trained models before sharing them on-chain. It is noteworthy to mention that studies on machine learning modeling techniques for blockchain-based phenotypic data analytics is very limited. This leaves a research gap

for researchers to explore in future.

5.8.4 Comparison and Evaluation of Blockchain-based Techniques for Phenotypic Data Sharing and Analytics

In this section, we discuss the comparison and evaluation of the techniques that leverage the features of the blockchain to collect, share and control access to phenotypic data as presented in Table 5.3. We evaluate these techniques based on (i) the security and privacy guarantees they provide, and (ii) the storage complexity of transactions within a blockchain network.

(1) Security and privacy: The security and privacy goals of blockchain-based techniques for recording transactions pertaining to phenotypic data are data integrity, provenance, access control and data privacy. Data integrity, provenance and access control are inherent properties of the blockchain and as a result all techniques presented Table 5.3 provide these features. However, the data privacy of patients should be protected since transactions recorded on the blockchain are public for the purposes of verifiability and auditability. This means blockchain transactions should not expose sensitive or confidential information about the patients and their phenotypic data. In literature, techniques have (i) encrypted patient sensitive information before recording them on the blockchain, or (ii) moved data storage to secure centralized off-chain storage nodes , or (iii) have utilized permissioned blockchains such as Quorum and Hyperledger Fabric which offers a variety of confidentiality mechanisms. In addition, the data privacy guarantees of a technique largely depends on the blockchain network architecture. For example, public blockchains such as Ethereum makes transactions publicly accessible while private and consortium blockchain networks such as Hyperledger Fabric which requires users to have public key infrastructure certificates are restricted and provides privacy for the data they store. We evaluate the data privacy guarantees of the techniques in Table 5.3 based on the confidentiality they provide for patients phenotypic data using cryptographic primitives and tools. We use symbols (i) ●: data privacy guarantee is met, (ii) ○: data privacy guarantee is not met, and (iii) N/A: data privacy is not available or not provided by authors. The techniques

[174, 175, 139, 142, 163, 197, 188, 172, 177, 170, 171, 176] encrypts patients phenotypic data with cryptographic algorithms such as symmetric and asymmetric encryptions to meet the data privacy requirement. These cryptographic algorithms provides provable security where the probability of breaking the scheme by a probabilistic polynomial time adversary is negligible. The techniques [173, 175, 142] have employed the permissioned nature of Hyperledger Fabric blockchain to manage patients data privacy. Other techniques have decoupled data storage from the blockchain by moving sensitive patients data to off-chain storage nodes such as InterPlanetary File System (IPFS), while recording encrypted pointers to the storage locations on IPFS [139, 172, 177]. This way transactions on the blockchain do not reveal references or pointers to where data is stored.

(2) Transaction storage complexity (TSC): Blockchain transactions are used to store and exchange phenotypic data, record request for granting and revoking access to data. This however creates blockchain scalability issues as transactions and the data they store grows (i.e. $\Psi_{tx} \propto \Psi_{bc}$, where Ψ_{tx} and Ψ_{bc} are the sizes of a transaction and the entire blockchain respectively). We compare and evaluate the storage cost of transactions generated by these techniques using asymptotic complexity (Big O notation) and present them in Table 5.3. We compute the complexities of these techniques as a function of the phenotypic data, metadata or pointers to storage locations of data they store, while omitting blockchain operational data such as nonce, transaction id and timestamp since they are constant for all schemes (i.e. $O(1)$). It is evident from Table 5.3 that the techniques [163, 140, 171, 177] have the least storage cost as they store data off-chain and record only lightweight reference pointers (e.g., URLs, IPFS hash) to storage locations on-chain, whereas techniques [173, 174, 175, 162, 170] that store the hash or encryption of patients data on-chain incurs the most storage cost. For techniques with the most storage cost, it is important to note that as the number of patients (n) and the blockchain nodes increases, the cost to transmit patients data between nodes will also increase hence leading to scalability issues. It still remains challenging to deal with blockchain scalability issues especially for schemes that store encrypted patients data on-chain. Encryption techniques that yield lightweight ci-

phertext could be used to encrypt patients data or data storage could be decoupled from the blockchain as proposed in [139].

5.9 Blockchain-based Techniques for Sharing and Processing Genomic Data

The traditional model of sharing genomic data where a middleman (i.e. third party such as a DTC) liaises between the genomic data owners (DOs) and data consumers (DCs) as shown Figure 5.4 faces challenges where DOs lose access and control of their data to middlemen, security and privacy concerns and DOs are not compensated for sharing their data. The decentralized, immutable, tamper-proof features of the blockchain with smart contracts have been utilized to address these challenges. In this section, we investigate and discuss the techniques that have been developed and proposed in literature based on the DLT genomic data sharing model to address the aforementioned challenges of the traditional model of sharing genomic data. For the sake of organization and clarity, we classify these techniques by their genomic application domain into secure collection and sharing of genomic data, and querying genomic data access logs on the blockchain. And finally, we evaluate and compare these techniques on the basis of storage, consent and compensation models, blockchain network type, security and query response time.

5.9.1 Secure Collection and Sharing of Genomic Data

During the collection and sharing of genomic data, one of the major concerns is enabling DOs to control access to their data. This will ensure that they have the ability to consent to sharing their data with certain DCs [200]. Controlling data access will empower DOs to consent to sharing their data based on terms and conditions of genomic data usage. With the emergence of the blockchain technology, the commercial space in genomics is pioneering the application of blockchain features to democratize genomic data sharing via DLT genomic marketplaces. This will allow DOs to own their data and freely control the terms and conditions on how DCs should use them. This has been demonstrated by genomic ecosystems such as Nebula Genomics [144],

Zenome [143], LunaDNA [201] and Shivom [182]. These ecosystems employ smart contracts to grant and revoke access, and to automate the terms and conditions of the consent agreement between DOs and DCs without third-party involvement. In addition, these ecosystems provide a genomic data discovery protocol for DCs to query genetic variants to determine DOs with genomic data of interest. Genomic data discovery protocols also include features to allow genomic data owners to be discovered by their phenotypic data such as disease conditions, medical history and allergies.

In addition to providing fine-grained access control for DOs genomic data, these ecosystems allow DOs to sell data access to DCs such as biomedical researchers, pharmaceutical and biotechnological companies. This encourages and incentivizes DOs to share genomic data while providing a vase and diverse genomic repository to advance research. While [144, 143, 182] pays and rewards DOs in cryptocurrency and genetic services for sharing data, [201] pays DOs in company shares. One limitation of compensating DOs for sharing data is that the reward value does not reflect the true value of the data [184].

Scientific literature on the application of blockchain features to collect and control access to genomic data while securing genetic variants and preserving the privacy of individuals is limited as it is a new research area. A recent study in this area proposed a framework to empower individuals to share their genomic data while enabling researchers to securely explore genomic datasets with controlled and transparent data access [145]. In this work, blockchain smart contracts are employed to develop access control policies to enforce individuals consent (i.e. dynamic consent) to share their genomic data and thereby eliminating the involvement of DTCs and the privacy risk they pose.

5.9.2 Querying Genomic Data Access Logs on the Blockchain

Genomic data collected and stored by different facilities and institutions needs to be queried and accessed across institutions for purposes such as collaborative biomedical research, drug discovery etc. Querying and accessing genomic data across multiple

Table 5.4: Comparison of blockchain functionality-based techniques for sharing and processing genomic data

Category	Scheme	Stor. Model	Cons. Model	CM	Crypt. Techn.	BC Arch.	BC Type	CA	SC	Data Priv.	Query Resp. Time
Secure collection and sharing of genomic data	Zenome, 2017 [143]	DOF	Dyn.	CT	SKE, PKC	N/A	ETH	FBA	✓	●	N/A
	Nebula, 2018 [144]	DOF	Dyn.	CT, GS	AES, SX, HE	PB	EXO	BFT	✓	●	N/A
	LunaDNA, 2018 [201]	COF	Dyn.	DI	SKE, PKC	N/A	N/A	N/A	N/A	●	N/A
	Shivom, 2018 [182]	COF	Dyn.	CT, GS	PKC, PRE	PB	HLF, ETH	PBFT	✓	●	N/A
	Grishin et al., 2021 [145]	COF	Dyn., broad	N/A	EC, SMC, DP	PB	EXO	BFT	✓	●	$O(n+q)$
Querying genomic data access logs on the blockchain	Özdayi et al., 2020 [165]	DON	N/A	-	N/A	PB	MLC	N/A	×	●	N/A
	Pattengale et al., 2020 [164]	DON	N/A	-	N/A	PB	MLC	N/A	×	●	SQ: $O(\gamma)$; MQ: $O(\lvert S\rvert \geq \gamma)$; RQ: $O(max(\gamma, S) + logi))$
	Gürsoy et al., 2020 [202]	DON	N/A	-	N/A	PB	MLC	N/A	×	●	SQ: $O(\gamma)$; MQ: $O(\gamma)$; RQ: $O(\gamma)$
	Ma et al., 2020 [203]	DON	N/A	-	N/A	PB	MLC	N/A	×	●	PQ: $O(t)$; MQ: $O(1)$; RQ: $O(\sum_{i=0}^{L} R_i/r_{L_i})$

Notations: **AES**: Advanced encryption standard; **Arch.**: Architecture; **BC**: Blockchain; **BFT**: Byzantine fault tolerance; **CA**: Consensus algorithm; **CM**: Compensation model; **CT**: Cryptocurrencies and tokens; **CB**: Consortium blockchain; **Cons.**: Consent; **Crypt.**: Cryptographic; **COF**: Centralized off-chain storage; **Dyn.**: Dynamic consent; **DOF**: Decentralized off-chain storage; **DON**: Decentralized on-chain storage; **DI**: Dividends; **DP**: Differential privacy; **ETH**: Ethereum; **EXO**: Exonum; **EC**: Elliptic Curve ElGamal encryption; **FBA**: Federated Byzantine Agreement; **GS**: Genetic services; **HE**: Homomorphic encryption; **HLF**: Hyperledger Fabric; **SX**: Intel Software Guard Extensions; **Priv.**: Privacy; **PoW**: Proof-of-work; **PoS**: Proof-of-stake; **PBFT**: Practical Byzantine fault tolerance; **PB**: Private blockchain; **UB**: Public blockchain; tolerance; **PRE**: Proxy re-encryption; **PKC**: Public key cryptography; **PQ**: Point query; **SC**: Smart contract; **Stor.**: Storage; **SMC**: Secure multi-party computation; **SKE**: Symmetric key encryption; **SQ**: Single constraint query; **Techn.**: Technique; **MQ**: Multiple constraint query; **MLC**: Multichain; **RQ**: Range query; **Resp.**: Response; **N/A**: Not available or applicable; $\boldsymbol{n}$: # of individuals; $\boldsymbol{q}$: # of query items (clinical and genetic criteria); $\boldsymbol{i}$: # of log lines inserted; $\boldsymbol{t}$: the size of the transaction IDs list; $\boldsymbol{\gamma}$: # of log lines returned by a query

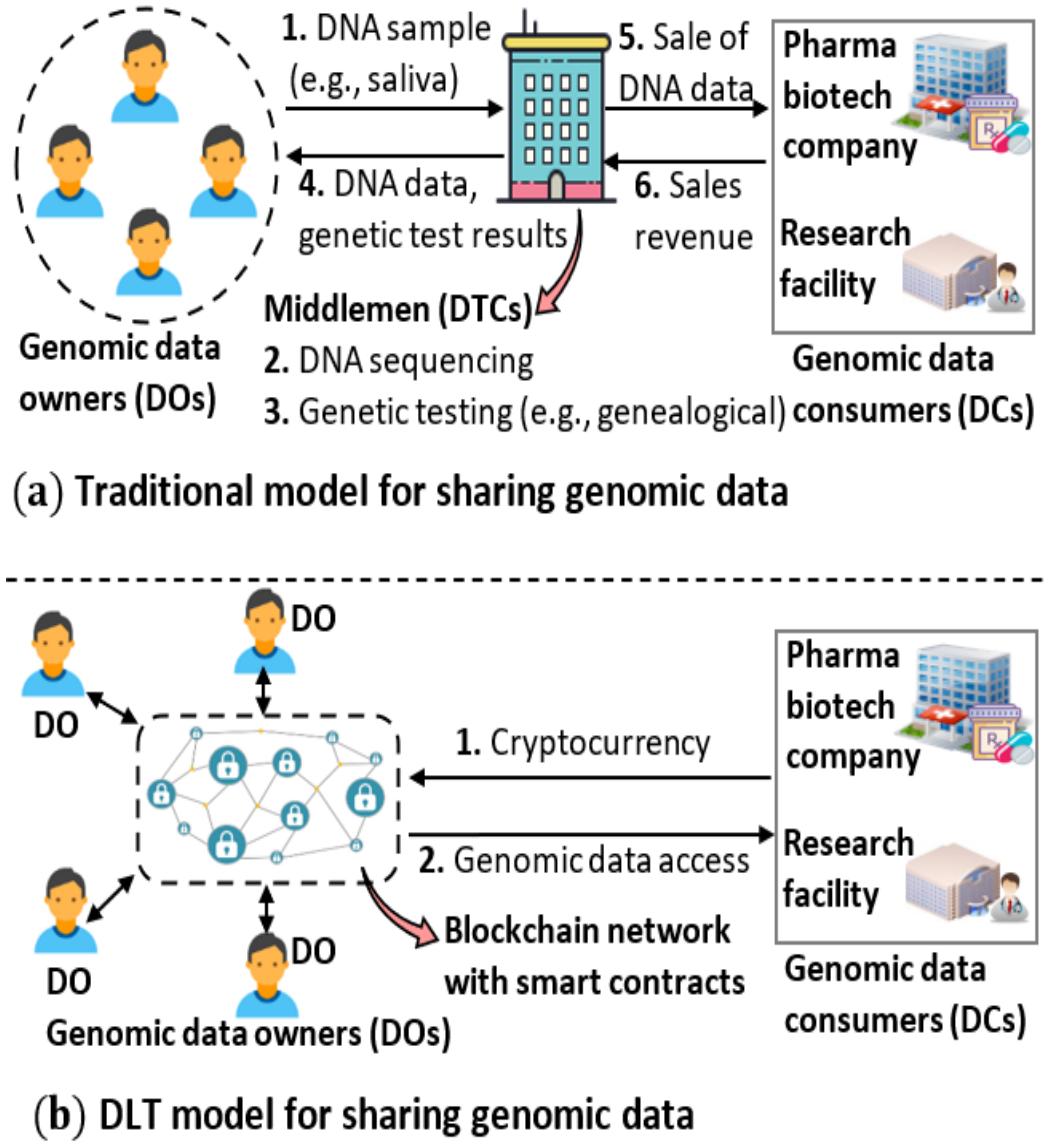


Figure 5.4: Traditional vs. DLT models for sharing genomic data. (**a**) Traditional model for sharing genomic data. Middlemen such as direct-to-consumer (DTC) genomic companies collect DNA data from individuals and sells them to pharmaceutical companies. This creates a problem where individuals lose control of their data. (**b**) DLT model for sharing genomic data. In this model, blockchain is used to empower genomic data owners to control their data, and sell data access to genomic data consumers. ***Notations:*** **DLT**: Distributed ledger technology; **DO**: Genomic data owner.

sites is currently challenged by auditing access activities such as requesting access to genomic datasets, viewing genomic dataset resources etc., in a transparent and secure manner. Blockchain has been proposed to address this challenge using its decentralized, auditable and tamper-proof nature. However, the data structure of blockchains does not provide an efficient way to query stored data on-chain. Recent studies have developed efficient techniques and data structures to log and query genomic data access logs across multiple sites from the blockchain [165, 164, 202, 203]. These studies

address complex query types (e.g., queries with single and multiple constraints, and range queries) over genomic access logs on the blockchain. While the studies [164, 203] are based on indexing techniques, the work in [165] utilizes data duplication and batch loading to run queries and the technique in [202] creates a data frame from the chains key-value instances which allows for efficient querying of logs.

5.9.3 Comparison and Evaluation of Blockchain-based Techniques for Sharing and Processing Genomic Data

In this section, we compare, discuss and evaluate the blockchain-based techniques that have been proposed in literature for sharing, accessing and processing genomic data in a secure manner as shown in Table 5.4. We discuss the comparison amongst these techniques based on security and query response time.

(1) Security and privacy: We evaluate the data privacy guarantees of the techniques in Table 5.4 based on the confidentiality they provide for DNA donors and genomic data such as sensitive variants. The genomic marketplaces and ecosystems [144, 143, 201, 182], and the technique [145] provide genomic data confidentiality as they encrypted data at rest with cryptographic techniques such as symmetric and asymmetric encryptions. In addition, the techniques for querying genomic access logs [165, 164, 202, 203] meet the security requirements as the logs they store and query do not contain any genetic sensitive information which can be used to link to the DNA donors identity.

(2) Query response time: Query response is the time taken to process a query and return results to the genomic data consumer (DC). This metric has been applied in literature to measure the query performance of techniques that provide a functionality to query genomic data access logs on the blockchain. As presented in Table 5.4, we compute and compare the query response time of techniques using the big O notation based on the size of the blockchain transactions they process, the number of log lines returned by a query etc. For single constraint queries (SQ), the query performance of techniques [164, 202, 203] are fairly the same as the response time which is linear to the number of log lines returned by a query (i.e. γ). The technique [203] has

the least response time for multiple constraint queries (MQ) as it is constant (i.e. $O(1)$). Finally, range queries (RQ) for the techniques [164, 202, 203] yield the most response time as the blockchain logs have to be processed to return a required interval of transaction timestamp and genomic dataset resource name.

5.10 Challenges and Future Directions of Blockchain Functionalities for Sharing and Processing Genomic and Phenotypic Data

In this section, we discuss the challenges and future directions on adopting and applying blockchain features for sharing and processing genomic and phenotypic data while protecting patients privacy.

5.10.1 Lack of Compliance with GDPR's Right to Data Erasure

The European Union's General Data Protection Regulation (GDPR) stipulates that data subjects have the right to request data controllers to delete their personal data from their platforms [204]. This gives EU residents, who have stored their personal phenotypic data on healthcare blockchains, the right to delete them. However, this contradicts with the very core nature of blockchains as they are designed to be immutable for purposes such as integrity checking and auditing. Current state-of-the-art techniques [173, 174, 175, 162, 170] that store patients genomic and phenotypic data (encrypted or in plaintext) on the decentralized blockchains fail to meet this regulation as blockchain records cannot be deleted. Challenging as it may seem, other techniques have proposed to store patient data off-chain, while recording hashes of patient data and pointers to data storage locations on-chain [205, 140]. This storage model enables patients to delete their data from off-chain storage nodes thereby complying with GDPR's right to data erasure. However, hashes of patient data and pointers to data storage locations are still left on-chain which poses a security risk. This remains a challenging and unexplored area. Researchers, as future direction, are exploring ways to make blockchain-based applications to comply with this regulation.

5.10.2 Blockchain Scalability Issues for Genomic and Phenotypic Datasets

State-of-the-art blockchain-based techniques for collecting and sharing genomic and phenotypic data are limited by scalability constraints inherent to the blockchain architecture such as transaction processing time per second (*tps*) [16]. In a real-life clinical settings, an application might generate more transactions (e.g., transactions to record phenotypic data access, retrievals, logging information etc.) that exceeds the blockchain's *tps* resulting in a backlog of transactions. This effect slows down the application. Transaction spacing and queueing techniques have been proposed by researchers to control the speed of sending out the transactions to the blockchain as a partial fix to this limitation [161]. However, this is still a largely unexplored area and as future work further research needs to be steered towards developing scalable blockchain-based solutions to acquire and share genomic and phenotypic data.

5.10.3 Tradeoff Between Genomic and Phenotypic Data Security and Blockchain Efficiency

The public blockchain has the feature of openness and transparency meaning that anyone can view transaction records and data stored on the public blockchain. Sensitive genomic and phenotypic data, information linking patients to their identities and pointers to off-chain storage locations stored on the blockchain should be encrypted to secure data and protect the privacy of patients. Encryption schemes come with their own complexity such as ciphertext expansions i.e., the size of the ciphertext is more than that of its corresponding plaintext [112]. The blockchain is constrained in storage capacity and will result in scalability issues if the size of the ciphertext stored on it grows. Thus, security parameters for encryption schemes should be chosen to provide desirable security levels while minimizing the ciphertext size to balance storage cost on the blockchain. On the other hand, permissioned enterprise blockchains such as Quorum and Hyperledger Fabric which offer a variety of confidentiality settings have been utilized in literature to protect genomic and phenotypic data [173, 175, 142].

However, the issue of blockchain scalability still remains a challenge as blockchains are not designed to store large volumes of data.

5.11 Conclusion

The traditional model of sharing and processing genomic and phenotypic data where third-party DTCs manage and control data presents challenges of access control, security and privacy, and lack of compensation for DNA donors. The emerging and growing popularity of blockchain technology comes with the features of decentralization, immutability, etc., to address these challenges. In this chapter, we present a comprehensive comparison on state-of-the-art techniques utilizing blockchain functionalities by investigating and classifying the storage, compensation and consent models employed to address the centralization, access control and lack of compensation issues of the traditional model. We further investigate and categorize state-of-the-art blockchain functionality-based techniques for sharing genomic and phenotypic data by their respective genomic and phenotypic application domain and blockchain functionalities. These techniques are compared and evaluated in terms of their security requirements, blockchain functionalities and transaction complexities. We believe the current trends and insights on the challenges, and future directions on the utilization of blockchain functionalities to share and process genomic and phenotypic data will serve as a guide for researchers in this field.

Chapter 6: Secure Federated Learning for DNA Sequence Classification with Verifiable Gradient Aggregation

Federated learning (FL) is an emerging technology that provides model training confidentiality in artificial intelligence by facilitating distributed learning across multiple parties while keeping their local training data private. Although promising, FL faces the following challenges: (i) security since an adversary can carry out attacks on shared gradients, (ii) how to verify the integrity of aggregated gradients from an FL parameter server, and (iii) degraded accuracy of models trained on heterogeneous structured data which are not independent and identically distributed (non-IID). These FL problems worsen when training a shared deep learning model to classify DNA sequences since the human genome is highly sensitive and heterogeneous. In this chapter, we propose a blockchain FL framework to address the challenges of training a shared model for DNA sequence classification. First, we propose a new technique to improve model accuracy by minimizing the weight divergence between the heterogeneous distributions of classes across multiple learning parties. Our technique is to create an initialization model trained on random DNA sequences which are bounded by the Shannon entropy range across all parties local DNA sequences. We then develop a cryptographic verifiable technique using bilinear pairing and homomorphic hash functions, and a new blockchain consensus algorithm (proof of aggregated gradients) to enable parties to verify the aggregated model gradients from the blockchain. Finally extensive experiments to train a model on the ChIP-seq dataset to predict DNA-binding motifs

NOTE: The content of this chapter has been submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence.*
Mohammed Yakubu, A., & Chen, Y. P. P. Secure Federated Learning for DNA Sequence Classification with Verifiable Gradient Aggregation. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* (Under Review).

demonstrates the improved accuracy and high efficiency of our proposed scheme.

6.1 Introduction

Deep learning has revolutionized artificial intelligence to analyze and learn from large volumes of data which traditional machine learning techniques cannot effectively cope with [206]. It has significantly improved the learning accuracies in different application domains such as natural language processing, DNA sequence classification and medical image analysis for cancer cell classification. Deep learning models achieve higher accuracy from learning from large-scale datasets. However, it is computationally expensive and time consuming for a single party to collect, annotate and train complex deep networks on large-scale datasets [207]. In addition, a model trained on a single party's local data is prone to overfitting thereby negatively affecting the performance of the model on new data. In recent years, federated learning (also called distributed deep learning) has been explored by researchers to address this problem by allowing a group of parties to collaboratively learn their collective data [26]. In federated learning (FL), multiple parties iteratively train their local data and upload intermediate local gradients to a central parameter server. The parameter server then aggregates the local gradients to update a shared model which is then downloaded by all parties for the next training round. This however poses four serious challenges for FL i.e. (i) **privacy issues**: the intermediate local gradients uploaded by parties to the parameter server can be exploited by an adversary to carry out attacks such as membership inference and model inversion attacks to infer sensitive information about the training datasets [208], (ii) **integrity of the aggregated gradients**: a dishonest parameter server, for reasons of personal gain, may return an incorrect updated model or drop some gradients from the aggregation process. Furthermore, parties find it challenging to verify the correctness of the updated model from the parameter server, (iii) **single-point-of-failure**: the parameter server which coordinates the training process is vulnerable to single-point-of-failure attacks. This means an attack on the parameter server will compromise the entire FL network, and (iv) **poor FL model accuracy**: existing work on FL assumes training datasets are evenly distributed over classes

across multiple parties i.e. independent and identically distributed (IID). However, this assumption of IID is unrealistic. In real life, training datasets are not independent and identically distributed (non-IID) over multiple parties. This heterogeneous nature of non-IID datasets degrades accuracy as a result of model weight divergence between the uneven distributions of classes amongst the learning parties [27]. The aforementioned challenges of FL are exacerbated when training a deep learning model on distributed or siloed genomic data to classify DNA sequences. This is a result of the following unique characteristics of the human genome compared to other data types such as image and text.

1. The human genome is highly sensitive and encodes information on an individual e.g., predisposition to diseases and family pedigree [109]. As a result, genomic data needs to be protected in accordance with regulations including the general data protection regulation (GDPR) and the health insurance portability and accountability act (HIPAA).

2. The human genome is highly diverse and heterogeneous as each individual's DNA is unique. This makes it more non-IID distributed amongst multiple genomic repositories. In addition, to increase accuracy in the non-IID setting, existing FL techniques [27] add a global data from a uniform distribution of random samples to parties local datasets, however this does not improve the accuracy of the models trained on genomic data. This is because the human genome is not a random distribution of DNA bases (A, G, T, C), but contains an average GC content of approximately 41% [209, 210].

Recent studies on FL techniques to improve the accuracy of deep learning models trained on non-IID datasets only show satisfactory improvements for image and natural language learning tasks [27, 28]. However, the models trained on highly heterogeneous biological datasets did not yield satisfactory accuracy improvements. In this chapter to address the challenges of FL for DNA sequence classification, we develop a blockchain FL scheme to improve the accuracy of a model trained on non-IID genomic data, while guaranteeing the confidentiality of local DNA sequences and

intermediate model gradients. Our solution eliminates single point of failure using blockchain decentralization and allows the parties to verify the correctness of the aggregated gradients. Our proposed technique is a generalized solution to improve the FL accuracy of a model trained on non-IID genomic data to accomplish DNA sequence classification task such as classifying viral genomes in human DNA samples, predicting transcription factor (TF) binding sites of DNA sequences etc.

Existing work on FL has adopted the distributed deep learning approach where a central server (mostly a cloud server) acts as the parameter server to coordinate the training process and the aggregation of intermediate gradients. The utilization of a central server to mediate the training process makes this approach vulnerable to single-point-of-failure attacks. For example, a recent study proposed a distributed deep learning framework to secure local gradients of parties with differential privacy techniques [211]. However, another study [212] proved that the work in [211] leaks local data information to an honest-but-curious server. This study further proposed a technique to address the privacy leakage of [211] by securing gradients with additive homomorphic encryption before sending them to the cloud server for gradient aggregation. Other studies to develop distributed deep learning frameworks utilized fully homomorphic encryption and symmetric encryptions techniques to protect gradients and model weight parameters on a cloud server [213, 214]. While [213] used fully homomorphic encryption, [214] adopted symmetric encryption. The limitations of [211, 212, 213, 214] are single point of failure, aggregated results from the cloud server are not verifiable and these studies do not address FL on non-IID datasets.

As a solution to address the single point of failure problem of distributed deep learning, recent studies have proposed decentralized federated learning frameworks based on blockchain technology [215, 216, 217, 207]. While [215, 216] use differential privacy to protect updated model parameters on the blockchain, [207] employs additive homomorphic encryption to facilitate gradients aggregation and [215] does not secure global models at all which raises privacy concerns. The drawbacks of the techniques in [215, 216, 217, 207] are that collaborating parties cannot verify the correctness of the aggregated gradients on the blockchain and they do not address the

problem of FL in non-IID settings.

The application of cryptographic verifiable computation techniques is of utmost importance in FL as it will enable collaborating parties to securely verify the computation work of gradient aggregation performed by the parameter server. After verifying the correctness of the aggregated gradient, a party can then update his local model for the next training round. This secure verifiability technique remains a major challenge in FL. A recent study, however, proposed a scheme to enable multiple parties to verify the correctness of aggregated gradients on a cloud server [218]. This study applied secret sharing to mask the parties local gradients from the cloud server and homomorphic hash functions to verify the computation performed by the cloud server. The scheme, however, suffers from the single point failure and the limitation of FL in non-IID settings.

Our proposed blockchain FL framework for learning non-IID genomic data for DNA sequence classification addresses the aforementioned limitations of current state-of-the-art privacy-preserving deep learning techniques. Specifically in our framework, parties encrypt their local gradients and upload them onto the blockchain for secure model aggregation. The aggregated model is then downloaded by all parties for the next training iteration. First, we propose a new technique to improve model accuracy by minimizing the weight divergence between the parties non-IID datasets. We protect the privacy of intermediate gradients with a threshold fully homomorphic encryption and develop a bilinear pairing and homomorphic hash function technique to verify the correctness of aggregated gradients from the blockchain. In summary, the contributions we make in this chapter are as follows:

- We propose a new technique to generate an initialization model trained on random DNA sequences which are quantified with Shannon's entropy from all parties local DNA sequences and contains the average human genome's GC content of approx. 41% [209, 210]. Our technique improves the accuracy of the shared model by minimizing the weight divergence between the heterogeneous distributions of classes amongst the learning parties.

- We design a cryptographic verifiable computation technique using bilinear pair-

ing and homomorphic hash functions to allow collaborating parties to verify the correctness of aggregated gradients from the blockchain.

- We develop a new blockchain consensus algorithm (proof of aggregated gradients) where a consensus committee verifies the model aggregation work of a lead validator prior to adding model update transactions to the blockchain ledger.

- We implement our proposed framework and demonstrate its performance on the ChIP-seq dataset in terms of model accuracy, computation cost and communication throughput. Our experiment results show that our scheme yields higher testing accuracy than current state-of-the-art techniques.

The remainder of this chapter is organized as follows. In Section 6.2, we present the preliminaries and cryptographic background required to understand our proposed framework. We discuss our system architecture in Section 6.3 and present our proposed framework in Section 6.4. Section 6.5 presents our security analysis. Our experiment setup and performance evaluation is present in Section 6.6. Finally, the chapter is concluded in Section 6.7.

6.2 Preliminaries and Background

In this section, we present the preliminaries on Shannon's information entropy and the cryptographic background required to understand our proposed framework.

6.2.1 Shannon's Information Entropy

In information theory, Shannon's entropy (uncertainty theory) has been applied to quantify the amount of information in a message or an event [219]. Originally Shannon's entropy was designed to encode, compress, and transmit data through a communication channel. In genomics, the DNA sequence contains inherent information which encodes the biological functionalities to produce proteins. The information encoded in DNA can be quantified using Shannon's entropy. It has been proven by recent studies that the Shannon's information of complete genomes is greater than that of random DNA sequences [220, 221]. In this work, we utilize Shannon's entropy to quantify the

information content of the local genomic data of each learning party. Specifically, we calculate the entropy of a DNA sequence S as $H(S) = -\sum_{i=1}^{n} \phi_i log(\phi_i)$, where n is the length of S and ϕ_i is the frequency of the ith DNA base in S.

6.2.2 Bilinear Pairing

A bilinear pairing is represented by the map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$; where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three multiplicative cyclic groups with the same prime order q. g and h are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. The bilinear pairing e has the following properties:

- **Bilinearity:** $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$; given random numbers $a, b \in \mathbb{Z}_q^*$, and $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$.
- **Non-degeneracy:** There exist a $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$ where $e(g, h) \neq 1$.
- **Computability:** For any $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, $e(g_1, g_2)$ can be computed efficiently.

6.2.3 Pseudorandom Functions

A pseudorandom function $PF_\Upsilon : \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbb{G}_1 \times \mathbb{G}_2$ with key $\Upsilon = (\Upsilon_1, \Upsilon_2)$ consists of two other pseudorandom functions $PF_{\Upsilon_1} : \{0,1\}^* \rightarrow \mathbb{Z}_q^2$ and $PF_{\Upsilon_2} : \{0,1\}^* \rightarrow \mathbb{Z}_q^2$. For inputs (I_1, I_2), $PF_{\Upsilon_1}(I_1) = (z_{I_1}, \upsilon_{I_1})$ and $PF_{\Upsilon_2}(I_2) = (z_{I_2}, \upsilon_{I_2})$, and consequently $PF_\Upsilon(I_1, I_2) = (g^{z_{I_1} z_{I_2} + \upsilon_{I_1} \upsilon_{I_2}}, h^{z_{I_1} z_{I_2} + \upsilon_{I_1} \upsilon_{I_2}})$.

6.2.4 Homomorphic Hash Functions

We define a collision-resistant homomorphic hash function [222] as $HF : R_q \rightarrow \mathbb{G}_1 \times \mathbb{G}_2$. For a BGV ciphertext μ, the homomorphic hash is computed as: $HF(\mu) = (\mathcal{M}, \mathcal{N}) = (g^{HF_{\tau,\kappa}(\mu)}, h^{HF_{\tau,\kappa}(\mu)}) \in \mathbb{G}_1 \times \mathbb{G}_2$; where $\tau \in R_q$ and $\kappa \in \mathbb{Z}_q$ are secret keys. We use homomorphic hash functions combined with pseudorandom functions and bilinear pairing to cryptographically verify the correctness of aggregated gradients during our consensus algorithm. Our approach to verify computation correctness extends the verifiable computation technique of [222]

6.2.5 Threshold Fully Homomorphic Encryption (TFHE)

Fully homomorphic encryption (FHE), is a cryptosystem which was first proposed in 2009 by Gentry to allow an unbounded number of computations on encrypted data [60]. The construction of FHE has been unpractical and inefficient for computation on encrypted data. However, recent advances based on ring learning with errors (RLWE) has led to the development of efficient variants of FHE such as leveled-FHE and somewhat homomorphic schemes [60]. Threshold fully homomorphic encryption (TFHE) is a multi-party construction of FHE which allows N parties to cooperatively decrypt a ciphertext without learning any information about their respective plaintext [223, 224]. In our proposed scheme, we use an efficient TFHE based on the FHE constructions of Brakerski, Gentry and Vaikuntanathan (BGV)'s cryptosystem [225, 226] whose security is based on the RLWE problem. This implementation of TFHE has an efficient computation and round complexity than the classic TFHE [224].

Key generation

Let the plaintext and ciphertext spaces be denoted by the polynomial rings $R_t = \mathbb{Z}_t[x]/\Omega_m(x)$ and $R_q = \mathbb{Z}_q[x]/\Omega_m(x)$ respectively where t and q are the plaintext and coefficient modulus respectively. $\Omega_m(x)$ is the mth cyclotomic polynomial of degree n in $\mathbb{Z}_t[x]$, and χ_0 and χ_1 are error distributions over R_q. Then for N parties, each party i's secret key sk_i is uniformly sampled from χ_0 as $sk_i \leftarrow \chi_0$ and public key is computed as $pk_i = (p, \sigma) = (\sigma.sk_i + e', \sigma)$; where $\sigma \leftarrow R_q^n$ and $e' \leftarrow \chi^n$. The public key pk for encryption is set as $pk = pk_1 + ... + pk_N$.

Encryption

The encryption of a message $m \in R_t$ by the encryption function $E(.)$ is given as: $E(m) = c$, where $c = (c_0 + c_1) = \big((\langle p, e' \rangle + m) \bmod q, \langle \sigma, e' \rangle \bmod q\big)$.

Share decryption

Each party i decrypts the ciphertext $c = (c_0 + c_1)$ with his secret key sk_i to obtain a decryption share using equation: $\mu_i = c_1.sk_i + 2e_0 \bmod q$; where μ_i is i's decryption

share and $e_0 \leftarrow \chi_0$.

Combining decryption shares

All N parties put their decryption shares $\mu_1, ..., \mu_N$ together to obtain the final decrypted message i.e. $\mu = c_0 + \sum_{i=1}^{N} \mu_i \bmod t$.

Homomorphic addition

The homomorphic addition of two ciphertexts c' and c'' is computed as: $HAdd(c', c'') = \big((c'_0 + c''_0) \bmod q, (c'_1 + c''_1) \bmod q\big)$.

In this work, we employ TFHE to allow a group of collaborating training parties to individually encrypt their local gradients before uploading them to the blockchain and collectively decrypting the aggregated gradients results from the blockchain. For brevity, in the rest of this work, we denote the ciphertext of the model weight w as $[w]$.

6.2.6 Blockchain

Blockchain is an emerging distributed ledger technology where transactions are immutable and stored in a decentralized ledger on a peer-to-peer (P2P) network [137]. Blocks in the ledger are chained using cryptographic hash pointers and each block contains transactions which are validated by nodes using a consensus algorithm. The structure of a block consists of a block header (containing hash values of the current and previous block, the timestamp etc.) and a block body which contains transactional records. In this work, a learning party stores model gradients in the metadata field of both upload and download transactions to and from the blockchain. In addition, the blockchain decentralized architecture eliminates single point of failure in our proposed framework.

6.3 System Architecture and Problem Formulation

This section presents our system and threat model and the problem formulation of FL non-IID datasets.

6.3.1 System Model

Our system model consists of the following entities as depicted in Figure 6.2.

- **Learning party (LP):** The learning party is an entity who wishes to perform a learning task on his private genomic data but is limited in data or computational power to carry out the entire training process. LP joins a collaborative group of parties with a common deep learning task to securely train a shared model on their collective training data.

- **Blockchain network (BC):** The blockchain is a decentralized network that permanently records the parties transactions pertaining to uploading and downloading gradients. The blockchain also homomorphically aggregates local gradients from the parties.

- **Validator (VA):** The validator is a node on the BC who collects, processes and validates blockchain transactions. Furthermore, the validator aggregates updated gradients from all parties which will be used for the next learning iteration.

- **Certified Key Manager (KM):** The KM initializes a collaborative learning group by generating public bilinear parameters, public key and secret keys for each LP. Then KM goes offline after this task.

6.3.2 Threat Model

Here we define the threat model we use in our proposed framework. In our threat model, we assume that the learning parties and validators are honest-but-curious [15] meaning that they might try to infer other parties private data, but will strictly execute the agreed protocol. In addition, our threat model meets the following requirements.

1. *Confidentiality of local gradients:* A learning party encrypts his local gradients before uploading them to the blockchain. Informally, the encryption guarantees the confidentiality of the parties local gradients. Other entities in our framework

should not learn any private information from the ciphertext or exploit it to carry out attacks such as membership inference and model inversion attacks [208].

2. *Verifiability of aggregated gradients on the blockchain:* Learning parties should be able to verify the correctness of aggregated gradients computed by the blockchain on their uploaded encrypted local gradients. This enables parties to ensure that the blockchain has not incorrectly updated the shared model or dropped some gradients from the aggregation process.

6.3.3 Federated Learning

Federated learning (FL) enables organizations which are constrained in data and compute resources to collaborate on learning a shared model for prediction, while keeping their local datasets private. At the end of the training process, the shared model learns from a wider and diverse range of data than what a single organization possesses in-house. In neural networks, the goal of training a model is to find the optimal parameters that minimize a loss function. For an FL problem with N parties this is given as: $\min_{w} \; g(w) = \sum_{i=1}^{N} \frac{d_i}{d}.g_i(w_i)$; where $g_i(w_i)$ is the loss of the prediction made with model parameters w, d_i is the number of samples of party i's local data and d is the total number of samples of all N parties. For each iteration t of the learning process, each party trains its local model w_i^t on its local data and uploads it to a parameter server. This is given as $w_i^t = w_i^{t-1} - \eta \nabla g_i(w_i)$; where η is the learning rate and $\nabla g_i(w_i)$ is the gradient of $g_i(w_i)$. The parameter server then aggregates the local models from all parties to get the global model $w_g^t = \sum_{i=1}^{N} \frac{d_i}{d} w_i^t$. The global model w_g^t is then used for the next iteration. This continues until the total number of iterations is reached.

Challenges of Non-IID Training Data Distribution

An FL training dataset can either be independent and identically distributed (IID) or not independent and identically distributed (non-IID). In the IID setting, the training dataset is evenly distributed over classes across multiple parties. However, it is unrealistic to assume that local training datasets will always be IID. The non-IID

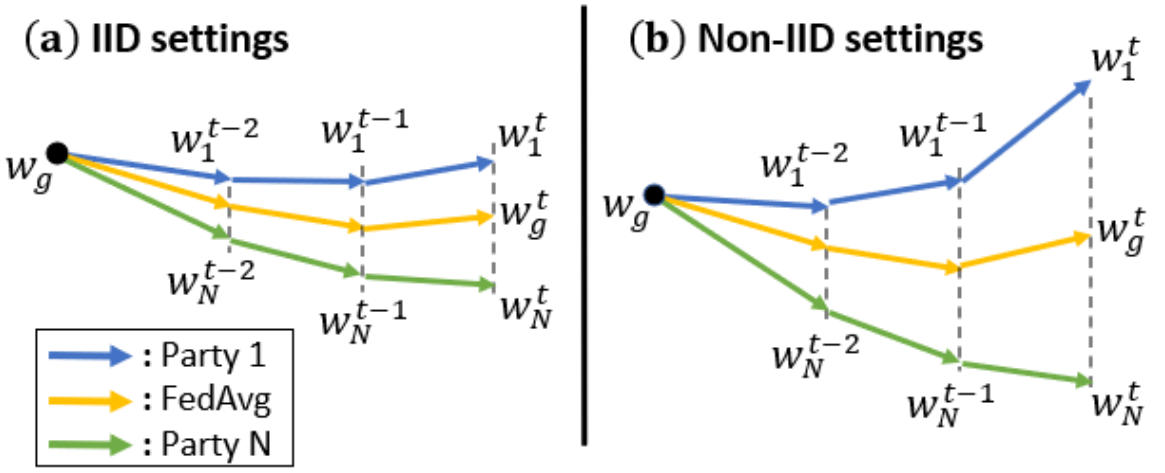


Figure 6.1: Comparison of model weight divergence in the IID and non-IID settings. **(a) IID setting:** Here, training datasets are evenly distributed over classes across multiple parties and the drift between the parties' local weights is close to the global optima. **(b) Non-IID setting:** The Parties' training datasets are statistically heterogeneous which results in a wide drift between their local weights and the global optima thereby degrading model accuracy.

setting is a more realistic case where the parties local data are statistically heterogeneous meaning that the data sizes and sample classes vary across multiple parties. This data heterogeneity degrades training accuracy in the non-IID setting compared to that of the IID setting. Training a model on non-IID datasets has been a major challenge in FL. As shown in Figure 6.1 for each party i ($i \in N$) when data is IID, the drift between the weights w_i^t is close to the global optima whereas for the non-IID setting, there is a wide drift between the local weights w_i^t and the global optima which increases with large numbers of local updates (i.e. local epochs). A recent study has proven that the accuracy reduction of a neural network trained on highly skewed non-IID data is a result of the weight divergence between the distributions over classes on each party's local data [27]. This study proposes a strategy to improve model accuracy by using a parameter server to create and distribute an initial subset of a global data amongst all parties. The global data is created from random samples uniformly distributed over classes contained in all parties. This ensures that the distance between the probability distributions over the parties classes is reduced, thereby increasing the model's accuracy. However, this approach cannot be applied to improve model accuracy on non-IID genomic data because of the following problems: (1) the parameter server requires the distribution of classes over each party's data in

order to create the global data. This exposes frequency patterns of each party's local data thereby posing a security risk. An adversary can exploit the DNA frequency patterns to carry out (i) model inversion attacks to extract sensitive information about the DNA sequence training dataset or (ii) membership inference attacks to breach an individual's privacy to determine whether his DNA sequence is part of the model's training dataset. (2) applying the technique proposed in [27] to create a global data from random DNA bases (A, G, T, C) will result in poor model accuracy. This is because the human genome is not a random distribution of DNA bases, but contains an average GC content of approximately 41%. In the next section, we propose a new technique using Shannon's entropy to improve the accuracy of a model trained to classify DNA sequences.

6.4 Our Proposed Framework

In this section, we present the technical implementation details of our proposed framework. Our proposed scheme as shown in Figure 6.2 addresses the core challenges of FL for DNA sequence classification which are (i) protecting the privacy of sensitive DNA sequences and local model gradients, (ii) improving model accuracy, and (iii) verifying the correctness of aggregated model gradients. We would like to note that our proposed technique is a generalized solution to improve the accuracy of FL across non-IID genomic datasets or repositories for DNA sequence classification. Examples of a DNA sequence classification task that can be accomplished are predicting the transcription factor (TF) binding sites of DNA sequences, classifying viral genomes in human DNA samples etc. In our proposed framework, we consider N parties $\mathcal{P}_1, ..., \mathcal{P}_N$ with sensitive local genomic datasets $D_{\mathcal{P}_1}, ..., D_{\mathcal{P}_1}$ respectively whose goal is to train a shared neural network model on their collective datasets without exposing their local DNA sequences and model gradients. To achieve this, our proposed framework comprises four phases: initialization of a collaborative learning group, generation of an initialization model, collaborative learning and a consensus algorithm. In the rest of this section, we present details of each phase.

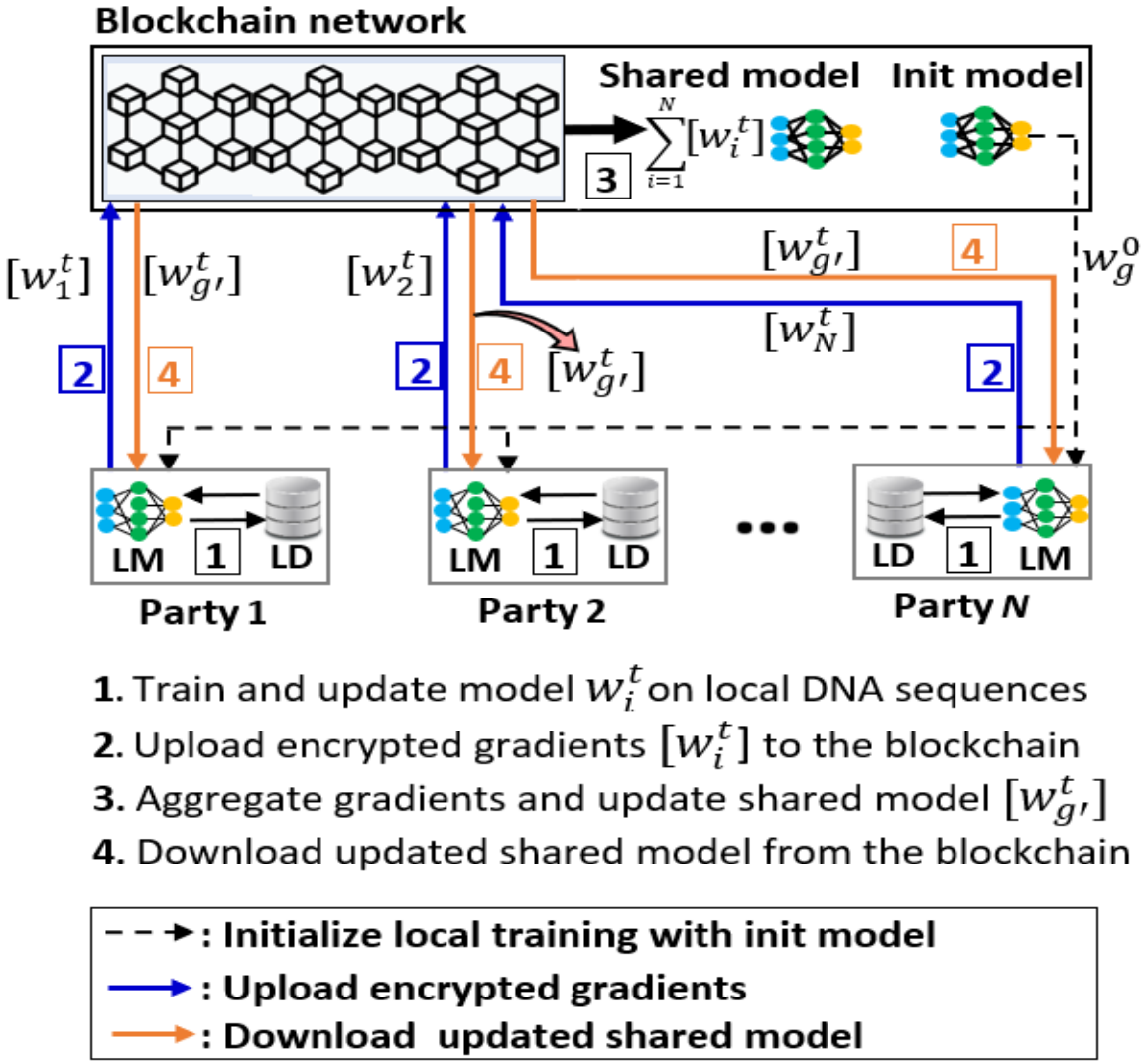


Figure 6.2: The system model of our secure collaborative learning on non-IID genomic data with verifiability. For each training iteration, parties train a local model on their local genomic data. Parties then encrypt and upload their model weights to the blockchain for aggregation of gradients to update the shared model. Finally, parties downloaded and collective decrypt the aggregated gradients for the next training iteration. ***Notes:*** **LM**: Local model; **LD**: Local genomic data.

6.4.1 Initialization of a Collaborative Learning Group

Assume a blockchain network with a genesis block has already been created, then each party $\mathcal{P}_i$ registers on the network to acquire a pseudo anonymous block address. Each party then writes a transaction to the blockchain to express a DNA deep learning intention that describes the type of DNA deep learning task he wishes to perform (e.g., DNA-TF binding sites etc.), the data type and data format. This enables parties with a common deep learning intention to form an FL collaborative group. Lets assume there are N parties within a collaborative group. The collaborative

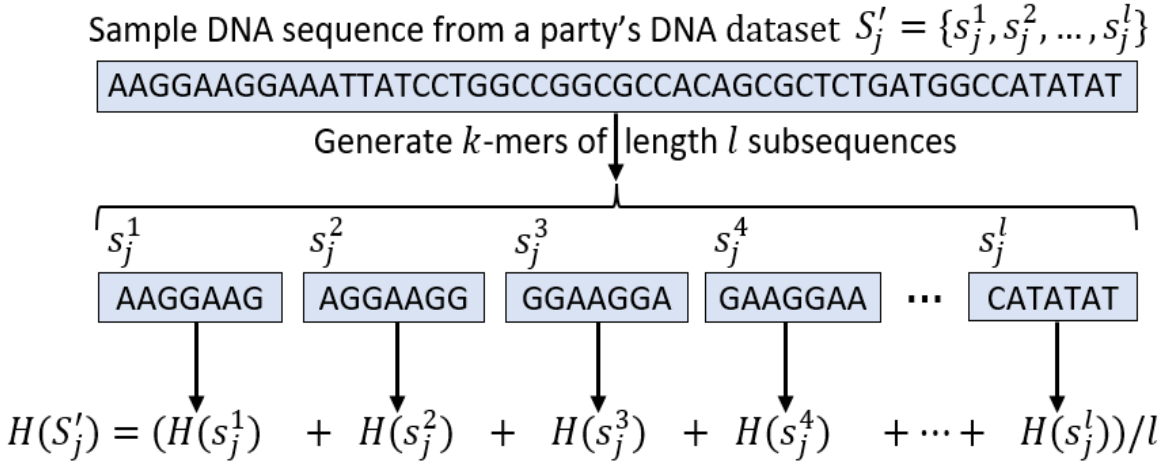


Figure 6.3: Calculation of Shannon entropy score of DNA sequence S'_j in dataset $D_{\mathcal{P}_i}$. The entropy of S'_j is computed as the average entropy of k-mer subsequences in S'_j (we use 7-mer for illustration purposes).

group is then initialized by a KM who generates (i) the public bilinear parameters $\xi = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, q, g, h)$, and (ii) the public key pk and the secret keys sk_i ($i \in (1, 2, ..., N)$) for TFHE. KM transmits $(\xi, pk, sk_i, \Upsilon = (\Upsilon_1, \Upsilon_2), (\tau, \kappa))$ to each party $\mathcal{P}_i$ in the collaborative group, where $\Upsilon = (\Upsilon_1, \Upsilon_2)$ and (τ, κ) are the secret keys for the pseudorandom functions and homomorphic hash functions respectively. KM then goes offline. Parties also agree on the settings and the configuration of the neural network such as total number of iterations ($\mathcal{T}$), number of epochs per iteration (ep), mini-batch size (bs), number of convolution layers (ly) and neurons per layer. These parameters are written to the blockchain.

6.4.2 Generation of Initialization Model Trained on Random DNA Sequences

In our proposed scheme, we address the accuracy degradation challenge of the non-IID setting in FL discussed in Section 6.3.3 by proposing a new technique to generate a global genomic dataset G within the information entropy range of the parties local DNA sequences and contains the average human GC content of approx. 41%. Our technique allows parties to quantify the amount of information contained in their local DNA sequences using Shannon's entropy measure. In doing so, the global data G can be generated to contain information from the distribution of all parties without actu-

ally knowing the DNA sequences stored in the parties' local data. The initialization model w_g^0 is then trained on G which is used by parties to initialize local training. The steps are as follows:

1. Elect party $\mathcal{P}_g$ to generate G: The collaborative group of N parties randomly selects a party $\mathcal{P}_g$ to generate G.

2. Generate k-mers of DNA sequences: Each party $\mathcal{P}_i$ has a local dataset $D_{\mathcal{P}_i}$ of n DNA sequences i.e. $D_{\mathcal{P}_i} = \{S_1, S_2, ..., S_n\}$. $\mathcal{P}_i$ computes the k-mer of each sequence S_j to get $S'_j \leftarrow S_j$ for $j \in (1, 2, ..., n)$; where S'_j is an array of k-mers of length k subsequences $S'_j = \{s_j^1, s_j^2, .., s_j^l\}$. This is shown in Figure 6.3.

3. Compute entropy of each sequence in $D_{\mathcal{P}_i}$: Estimate the Shannon entropy score of each sequence S'_j as the average entropy of k-mer subsequences in S'_j.

$$H(S'_j) = \frac{\sum_{m=1}^{l} H(s_j^m)}{l} \tag{6.1}$$

$$\forall s_j^m \in \{s_j^1, s_j^2, .., s_j^l\};\ H(s_j^m) = -\sum_{i=1}^{k} \phi_i log(\phi_i)$$

where s_j^m is the mth subsequence of S'_j, k is the length of s_j^m and ϕ_i is the frequency of the ith DNA base in s_j^m. At the end of this step, each party estimates the information content of $D_{\mathcal{P}_i}$ as $H(D_{\mathcal{P}_i}) = H(S_j) \leftarrow H(S'_j)$ for $j \in (1, 2, ..., n)$. All parties then submit their data entropy range $\big(H(D_{\mathcal{P}_i})_{min}, H(D_{\mathcal{P}_i})_{max}\big)$ to $\mathcal{P}_g$ where $H(D_{\mathcal{P}_i})_{min}$ and $H(D_{\mathcal{P}_i})_{max}$ are the minimum and maximum entropies of $D_{\mathcal{P}_i}$ respectively.

4. Random generation of G: Party $\mathcal{P}_g$ generates DNA sequences from random samples of DNA bases (A,G,T,C) with GC content of approx. 41% and information content bounded by $H(D)_{min}$ and $H(D)_{max}$, where $H(D)_{min}$ and $H(D)_{max}$ are the minimum and maximum information entropies across all N parties.

5. Train the initialization model w_g^0 : $\mathcal{P}_g$ retrieves the learning parameters from the blockchain that were agreed upon by all parties and trains the initialization model w_g^0 on G. $\mathcal{P}_g$ then writes w_g^0 to a blockchain transaction $Txn_{initmodel}$. The model w_g^0 is used by all parties to initialize training on their local datasets. Since w_g^0 is trained on a global data distribution across all N parties, the EMD between the distributions

Algorithm 7: *Collaborative learning of a shared model across N parties for DNA sequence classification* **(CLA)**

1. UPDATING GLOBAL SHARED MODEL
   (a) Create shared model object w_g
   (b) Train shared model w_g in $\mathcal{T}$ iterations on parties' local data
   for $t = 0, 1, 2, ..., \mathcal{T} - 1$ **do**
      if $t = 0$ **then**
         Get initialization model w_g^0 from blockchain
         $w_g \leftarrow w_g^0$; $w_g^0 \leftarrow Txn_{initGmodel}$
      foreach *party* $\mathcal{P}_i$ $i, \in N$ ***in parallel*** **do**
         LocalTraining(w_g)
      (i) Execute consensus algorithm (presented in Section 6.4.4) to aggregate local gradients $[w_{g'}^t] = \sum_{i=1}^{N} [w_i^t]$ and verify computation.
      (ii) Parties collaboratively decrypt $[w_{g'}^t]$ to obtain $w_{g'}^t$
      (iii) Update shared global model for the next iteration $w_g \leftarrow w_{g'}^t$
2. LOCAL TRAINING **LocalTraining**$(w_{g'}^t)$
   (a) Compute the average of the updated global weights $w_g^t = \frac{d_i}{d} w_{g'}^t$
   (b) Train local model on local data $D_{\mathcal{P}_i}$
   for *local epoch* $le = 1, 2, ..., ep$ **do**
      foreach *batch bs of* $D_{\mathcal{P}_i}$ **do**
         $w_i^t = w_g^t - \eta \nabla g_i(w_g^t, b)$
   (c) Encrypt w_i^t, and compute authentication tag and witness on $[w_i^t]$
      $[w_i^t] = E(w_i^t)$
      $\psi_i, \Lambda_f =$ **AuthWit**$(i, N, \varrho, [w_i^t], \xi, \Upsilon, (\tau, \kappa))$
   (d) Upload updated model parameters to blockchain via gradient upload transaction
      $Txn_{i,upload} = \left\{ [w_i^t], \psi_i \right\}$

over classes on each party will be reduced, thereby improving the accuracy in the non-IID setting [27].

6.4.3 Collaborative Learning for DNA Sequence Classification

In this section, we develop protocols for parties within an FL collaborative group to securely train a shared model on their local genomic datasets to classify DNA sequences. Figure 6.4 shows the workflow of our framework for DNA sequence classification. Parties collaboratively train the shared model in $\mathcal{T}$ iterations. In each iteration t, parties encrypt and upload their updated local gradients to the blockchain. The blockchain then securely aggregates these gradients to get an updated global model for the next round as presented in Figure 6.4. To start collaborative learning, each party $\mathcal{P}_i$ acquires the initialization model weights w_g^0 (generated in Section 6.4.2) from the blockchain to initialize local training on their local dataset $D_{\mathcal{P}_i}$. Below are the

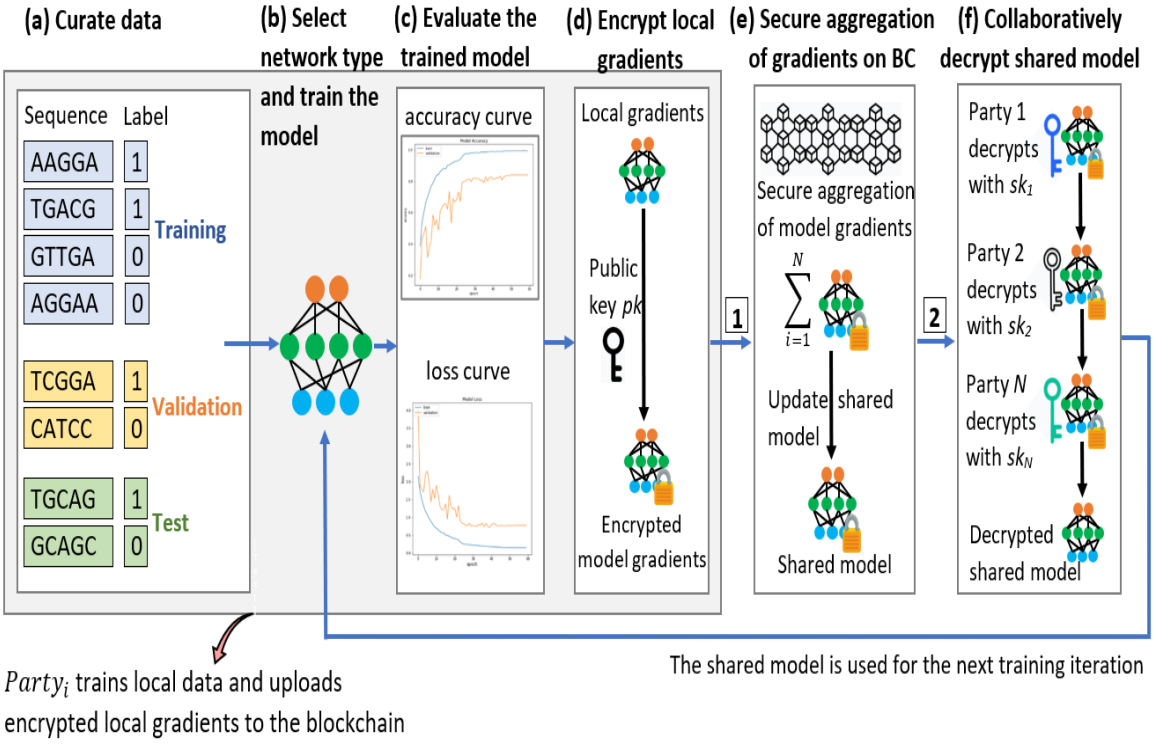


Figure 6.4: The workflow of our framework for DNA sequence classification. **(a) Curate data:** The local DNA dataset at party i is randomly split into training, validation, and testing sets. **(b) Select network type and train the model:** The appropriate neural network architecture is selected and configured in terms of total number of iterations, number of epochs per iteration, mini-batch size, convolution layers etc. The neural network is then trained on the training dataset. **(c) Evaluate the trained model:** The trained model is evaluated against the validation and test datasets using metrics such as accuracy, loss etc. **(d) Encrypt local gradients:** The local gradients are encrypted and uploaded to the BC. **(e) Secure aggregation of gradients on BC:** The BC aggregates the gradients from parties to update the shared model. **(f) Collaboratively decrypt the shared model:** Learning parties collaboratively decrypt the shared model using their secret keys.

steps for each iteration t of the shared model learning process:

Step 1: Each party $\mathcal{P}_i$ trains his local model on local data $D_{\mathcal{P}_i}$. i.e. $w_i^t = w_g - \eta \nabla g_i(w_g, b)$; where b is the minibatch size of $D_{\mathcal{P}_i}$. If $t = 0$ (i.e. initialization) $w_g = w_g^0$, else $w_g = w_g^t$, where w_g^t is the updated global model weights.

Step 2: $\mathcal{P}_i$ then encrypts w_i^t with encryption technique in Section 6.2.5 to get $[w_i^t]$ and computes an authentication tag and witness on $[w_i^t]$ using **Algorithm 8** i.e. $\psi_i, \Lambda_f = \textbf{AuthWit}(i, N, \varrho, [w_i^t], \xi, \Upsilon, (\tau, \kappa))$, where $\psi_i = (\mathcal{M}_i, \mathcal{N}_i, \mathcal{Q}_i, \mathcal{R}_i, \Theta_i = 1)$. The authentication tag ψ_i and witness Λ_f will be used to verify the correctness of

the aggregated model parameters to guarantee the integrity of the computed results. $\mathcal{P}_i$ then uploads $[w_i^t]$ and ψ_i to the blockchain via gradient upload transaction $Txn_{i,upload} = \left\{[w_i^t], \psi_i\right\}$

Step 3: After all parties have uploaded their local gradients for iteration t, the consensus algorithm (presented in Section 6.4.4) is executed to aggregate the model weights $[w_i^t]$ using homomorphic summation i.e. $[w_{g'}^t] = \sum_{i=1}^{N}[w_i^t]$. If the aggregation computation work is verified by the consensus committee then a block with a transaction containing $[w_{g'}^t]$ is written to the blockchain.

Step 4: Parties collaboratively decrypt $[w_{g'}^t]$ to obtain $w_{g'}^t$ using their secret keys sk_i as presented in Section 6.2.5.

Step 5: Finally for iteration t, each party $\mathcal{P}_i$ computes the average of the summed global weights $w_{g'}^t$ to obtain: $w_g^t = \frac{d_i}{d} w_{g'}^t$, where w_g^t is the updated global model to be used for local training in the next iteration, d_i is the number of DNA samples in $\mathcal{P}_i$ local dataset and d is the total number of DNA samples of all parties. Step 1 is repeated until the maximum number of iteration is reached or the shared model w_g has converged. We remark that at the end of this protocol no party's sensitive DNA sequences is revealed during local and global model parameter updates and computation.

We would like to note that our collaborative decryption scheme can be seen as an N-out-of-N scheme, meaning that all parties are required to participate in the decryption process. We acknowledge that this comes with a limitation in decryption when some parties drop-out of the training process. As a future work, our framework can be extended to implement a ε-out-of-N decryption scheme using secret sharing techniques to allow any subset of ε parties to perform decryption.

6.4.4 Consensus Algorithm (Proof of Gradient Aggregation)

Blockchain consensus algorithms based on traditional protocols such as Proof-of-work (PoW) requires high computational resources to select a node to write a block to the network. In addition, such consensus protocols are not developed to verify gradient aggregation operations. To address this problem, we propose a consensus protocol,

Algorithm 8: *Compute authentication tag of encrypted model weights and a witness for the model aggregation operation* **(AuthWit)**

Input: $i \in N, N, \varrho =$ "*aggregate*", $[w^t], \xi, \Upsilon = (\Upsilon_1, \Upsilon_2), (\tau, \kappa)$
Output: $\psi_i = (\mathcal{M}_i, \mathcal{N}_i, \mathcal{Q}_i, \mathcal{R}_i, \Theta_i = 1), \Lambda_f$

1. Compute authentication tag of the encrypted model weights $[w^t]$.
   (a) Compute the homomorphic hash of $[w^t]$.
   $(\mathcal{M}_i, \mathcal{N}_i) \leftarrow HF([w^t]) = (g^{HF_{\tau,\kappa}([w^t])}, h^{HF_{\tau,\kappa}([w^t])})$
   (b) Compute pseudorandom function on i and ϱ where $\alpha \in \mathbb{Z}_q$.
   $(\mathcal{Q}'_i, \mathcal{R}'_i) \leftarrow PF_\Upsilon(i, \varrho) = (g^{z_i z + \upsilon_i \upsilon}, h^{z_i z + \upsilon_i \upsilon})$
   $\mathcal{Q}_i \leftarrow (\mathcal{Q}'_i, \mathcal{M}_i^{-1})^{1/\alpha} = (g^{z_i z' + \upsilon_i \upsilon' - HF_{\tau,\kappa}([w^t])})^{1/\alpha}$
   $\mathcal{R}_i \leftarrow (\mathcal{R}'_i, \mathcal{N}_i^{-1})^{1/\alpha} = (h^{z_i z' + \upsilon_i \upsilon' - HF_{\tau,\kappa}([w^t])})^{1/\alpha}$
   (c) Set authentication tag $\psi_i = (\mathcal{M}_i, \mathcal{N}_i, \mathcal{Q}_i, \mathcal{R}_i, \Theta_i = 1)$.
2. Compute a verification information and witness on $f(.)$ for $\varrho =$ "*aggregate*" where $f(.)$ is a function for $f(x) = \sum_{i=1}^{N} x_i$
   (a) Calculate $(z_i, \upsilon_i) \leftarrow PF_{\Upsilon_1}(i)$.
   (b) Interpret (z_i, υ_i) as a linear form φ_i that maps (y_1, y_2) to $\varphi_i(y_1, y_2) = (z_i y_1, y_2 \upsilon_i)$, where $y_1, y_2 \in \mathbb{Z}_q$.
   (c) Compute $\varphi \leftarrow f(\varphi_1, ..., \varphi_N)$ i.e. $\varphi(y_1, y_2) = \sum_{i=1}^{N} \varphi_i(y_1, y_2)$.
   (d) Set witness on $f(.)$ as $\Lambda_f \leftarrow \Lambda(y_1, y_2) = \varphi(y_1, y_2)$.
3. Return authentication tag ψ_i and witness Λ_f.

proof of gradient aggregation (PoGA) which is based on the work in [227]. Our PoGA is specific to our FL problem to validate the computation work of gradient aggregation. First, for each iteration a leader $\mathcal{V}^t_{leader}$ is randomly selected from the validators using the cryptographic sorting technique of [227] and parties who are collaboratively training a shared model form a consensus committee to verify transactions and validate the computational work of $\mathcal{V}^t_{leader}$. For each learning iteration, below are the steps of our consensus protocol:

Step 1: The leader $\mathcal{V}^t_{leader}$ gathers gradient upload transactions $Txn_{i,upload}$ (for $i \in N$) and aggregates the model weights $[w^t_i]$ using homomorphic summation i.e., $[w^t_{g'}] = \sum_{i=1}^{N} [w^t_i]$. $\mathcal{V}^t_{leader}$ then computes the proof of the aggregated result $[w^t_{g'}]$ to get $Proof^t_{g'} = (\mathcal{M}, \mathcal{N}, \mathcal{Q}, \mathcal{R}, \Theta)$. This is given as: $\mathcal{M} = \prod_{i=1}^{N} \mathcal{M}_i$, $\mathcal{N} = \prod_{i=1}^{N} \mathcal{N}_i$, $\mathcal{Q} = \prod_{i=1}^{N} \mathcal{Q}_i$, $\mathcal{R} = \prod_{i=1}^{N} \mathcal{R}_i$, $\Theta = \prod_{i=1}^{N} \Theta_i$. Next, $\mathcal{V}^t_{leader}$ packs $Txn_{i,upload}, [w^t_{g'}]$ and $Proof^t_{g'}$ into a block BL_t and broadcast it to the committee for verification.

Step 2: The committee reach a consensus on the state of the block BL_t by executing Byzantine agreement protocol. Each committee member verifies the transactions of BL_t and the computation work of $\mathcal{V}^t_{leader}$. The computation work of $\mathcal{V}^t_{leader}$ is verified against $Proof^t_{g'}$ using **Algorithm 9** i.e., $acc =$ **VerifyComp**$([w^t_{g'}], Proof^t_{g'}, \Lambda_f, \varrho, \xi, \Upsilon, (\tau, \kappa))$. Based on the rule of majority voting [227], if BL_t is verified and approved by more

Algorithm 9: *Verification of computation work to aggregate model weights* **(VerifyComp)**

Input: $[w^t_{g'}], Proof^t_{g'} = (\mathcal{M}, \mathcal{N}, \mathcal{Q}, \mathcal{R}, \Theta), \Lambda_f, \varrho =$ "*aggregate*", ξ, $\Upsilon = (\Upsilon_1, \Upsilon_2), (\tau, \kappa)$

Output: 1 (accept) or 0 (reject).

1. Compute $(z, \upsilon) \leftarrow PF_{\Upsilon_2}(\varrho)$
2. Compute Λ_f on (z, υ) i.e. $\Lambda(z, \upsilon) = \varphi(z, \upsilon)$
   $\varphi \leftarrow \varphi(z, \upsilon) = \sum_{i=1}^{N}(z_i z + \upsilon_i \upsilon)$
3. Calculate $W \leftarrow e(g, h)^{\varphi}$ and $(\mathcal{M}', \mathcal{N}') \leftarrow HF([w^t_{g'}])$
2. Verify if the following equations hold.
   $(\mathcal{M}, \mathcal{N}) \stackrel{?}{=} (\mathcal{M}', \mathcal{N}');\ e(\mathcal{M}, h) \stackrel{?}{=} e(g, \mathcal{N});\ e(\mathcal{Q}, h) \stackrel{?}{=} e(g, \mathcal{R})$
   $W \stackrel{?}{=} e(\mathcal{M}, h).e(\mathcal{Q}, h)^{\alpha}$
   if *any of the equations do not hold* **then**
   return 0 (reject)
   else
   return 1 (accept)

than 2/3 of the committee members (i.e., acc = accept), then the block signed with $\mathcal{V}^t_{leader}$ digital signature is appended to the blockchain, and $\mathcal{V}^t_{leader}$ earns block reward. On the other hand, if verification fails, the block BL_t is rejected and a new leader is selected who starts from Step 1. The more often a validator's block is rejected, the lower the probability of choosing that validator as a leader to add a block.

6.5 Security Analysis

In this section, we demonstrate that the security and confidentiality of the shared global model trained by our proposed collaborative learning protocol **CLA** is guaranteed. We then analyze the correctness of the aggregated model weights returned by the validator $\mathcal{V}^t_{leader}$.

6.5.1 Security of the Trained Model

We first present a formal definition of semantic security for the BGV cryptosystem and then analyze the security of the trained shared global model.

Definition 3 *(Semantic Security, IND-CPA [15]). The BGV scheme is semantically secure if for any probabilistic polynomial-time (PPT) adversary* $\mathcal{A}$, $\big|Pr[\mathcal{A}(pk, E(pk, w_0)) = 1] - Pr[\mathcal{A}(pk, E(pk, w_1)) = 1]\big|$ *is negligible in the security parameter* λ *where* w_0 *and* w_1 *are chosen by* $\mathcal{A}$*. This indicates that the probability of distinguishing* w_0 *or* w_1 *by* $\mathcal{A}$ *is negligible.*

Theorem 6 *If the underlying BGV encryption scheme is semantically secure, then the confidentiality of the training parties gradients uploaded to the blockchain is guaranteed.*

Proof 7 *We use the following game to prove this theorem. Suppose an adversary $\mathcal{A}$ with non-negligible advantage performs the following attack:*

1. *A challenger $\mathcal{C}$ generates (pk, sk) and gives pk to $\mathcal{A}$*
2. *$\mathcal{A}$ randomly generates two model weights w_0 and w_1 of equal length and sends them to $\mathcal{C}$.*
3. *$\mathcal{C}$ randomly flips a coin γ (i.e. $\gamma \leftarrow \{0, 1\}$) and encrypts w_γ to get $[w_\gamma]$. $\mathcal{C}$ then sends $[w_\gamma]$ to $\mathcal{A}$.*
4. *Finally, $\mathcal{A}$ guesses $\gamma' \in (0, 1)$ for γ and wins the game if $\gamma = \gamma'$.*

The security of our scheme relies on the semantic security of BGV used to encrypt model weights which ensures indistinguishability under chosen-plaintext attack (IND-CPA) if the Ring-Learning With Error (RLWE) problem is hard. The ciphertext $[w_0]$ and $[w_1]$ are pseudo-random and the probability that $\mathcal{A}$ guesses a valid γ' for $\gamma = \gamma'$ to distinguish between the encryptions of w_0 and w_1 is negligible.

6.5.2 Correctness of Verification for Aggregated Model Weights

Here, we analyze the correctness of the verification **(VerifyComp)** performed by a learning party to verify the computation work of the validator $\mathcal{V}^t_{leader}$ to aggregate model weights during the consensus algorithm. We present below Theorem 7 to prove the correctness of the verification.

Theorem 7 *The verification of the correctness of the aggregated model weights $[w^t_{g'}]$ by a learning party belonging to a consensus committee verifies that $[w^t_{g'}]$ is indeed the aggregated result of $\sum_{i=1}^{N}[w^t_i]$ if $W = e(\mathcal{M}, h).e(\mathcal{Q}, h)^\alpha$, $e(\mathcal{Q}, h) = e(g, \mathcal{R})$ and $e(\mathcal{M}, h) = e(g, \mathcal{N})$.*

Proof 8 *We start by showing that each party can correctly verify* $(\mathcal{M}, \mathcal{N})$. *The party receives* $[w_{g'}^t]$ *and* $Proof_{g'}^t = (\mathcal{M}, \mathcal{N}, \mathcal{Q}, \mathcal{R}, \Theta)$ *from the validator* $\mathcal{V}_{leader}^t$. *The party computes* $W = e(g,h)^{\varphi} = e(g,h)^{\sum_{i=1}^{N}(z_i z + \upsilon_i \upsilon)}$ *and checks whether* $W = e(\mathcal{M}, h).e(\mathcal{Q}, h)^{\alpha}$. *According to the l-bilinear Diffie-Hellman inversion (l-BDHI) assumption [222],* $W = e(\mathcal{M}, h).e(\mathcal{Q}, h)^{\alpha}$ *holds only when* $\mathcal{Q}$ *contains* $\sum_{i=1}^{N}(z_i z + \upsilon_i \upsilon)$ *within the exponent of* g. *Each party can deduce that* $\mathcal{Q} = \prod_{i=1}^{N} \mathcal{Q}_i = g^{\sum_{i=1}^{N}(z_i z + \upsilon_i \upsilon) - HF_{\tau,\kappa}([w_{g'}^t])}$, *and hence knows that* $\mathcal{M} = g^{HF_{\tau,\kappa}([w_{g'}^t])}$. *Next based on the Decisional Diffie-Hellman (DDH) assumption [222], if each party checks that* $e(\mathcal{Q}, h) = e(g, \mathcal{R})$ *and* $e(\mathcal{M}, h) = e(g, \mathcal{N})$ *are true, then the party is sure that the validator has correctly computed* $\mathcal{R}$ *and* $\mathcal{N}$, *and consequently verifies the correctness of* $(\mathcal{M}, \mathcal{N})$. *Finally, if* $HF([w_{g'}^t]) = (\mathcal{M}, \mathcal{N})$ *is true, then the party is convinced that* $[w_{g'}^t]$ *is indeed the aggregated result of* $\sum_{i=1}^{N}[w_i^t]$.

6.5.3 Security of our proposed framework

In this section, we discuss the security of our proposed framework. Our proposed framework meets the security requirements presented in Section 6.3.2. First, the confidentiality of the local gradients is met since they are encrypted with BGV cryptosystem which is semantically secure as proven in Theorem 6. In addition, the model aggregation performed on the blockchain network is homomorphically computed and does not expose sensitive information about the DNA sequences. Finally, as shown in Theorem 7, our scheme enables parties to verify the correctness of the aggregated gradients computed on the blockchain network. This enables parties to check that the aggregated gradients computed by the blockchain has not been tampered with.

6.6 Experiment Evaluation

In this section, we implement and evaluate the performance of our proposed framework. We use Python and Tensorflow to build our deep learning environment on a PC with 2.60 GHz processor (4 cores) and 16 GB memory. We use the convolutional neural network (CNN) architecture of [228] to discover DNA-binding motifs. In addition, for local training we set the learning rate to 0.01, mini-batch size to 32 and number of epochs per iteration to 1. We implement our threshold fully homomorphic

encryption with the PALISADE library [229] which is an open-source lattice cryptography software library and bilinear pairing with the JPBC library [230]. For the BGV cryptosystem, we set the cryptosystem parameters to achieve a 128 bit security level. We set the number of learning parties to 10 (i.e. $N = 10$) and the number of learning iterations to 100. We collaboratively train the CNN model on ChIP-seq datasets from the ENCODE project [231] to predict whether a DNA sequence binds to any transcription factor (TF). We selected 70,000 top-ranking DNA sequences from records in the peak files of the ChIP-seq datasets as positive sequences where each sequence consists of 102 base pairs. A positive DNA sequence is a transcription factor binding site whereas a negative one is not. We generate the negative sequences by shuffling the positive sequences with matching dinucleotide composition. It is worth noting that the same ChIP-seq dataset preparation techniques have been applied to accomplish the deep learning task in [232]. This results in 140,000 DNA sequences from which we use 65,000 as training samples, 5,000 for validation and 10,000 as test samples. We split the training set between the 10 parties for different non-IID data distribution levels $\rho = 0.9, 0.8, 0.7, 0.6, 0.5$, where $\rho = 0.9$ means that 90% of each party's training data belongs to one class and 10% belongs to the other class. $\rho = 0.8$ means that 80% of the data belongs to one class and 20% belongs to the other class. $\rho = 0.7$ means 70% of data belongs to one class and the remaining 30% belongs to the other class, and so on for $\rho = 0.6$ and 0.5. We then evaluate the efficiency and performance of our proposed scheme in a multi-party setting in terms of testing accuracy, computation cost and communication throughput.

6.6.1 Testing Accuracy Evaluation

In this section, we evaluate the testing accuracy of the shared model. We train the shared model to predict the DNA sequence TF binding sites with our proposed scheme **CLA** and the current state-of-the-art FL schemes **FedAvg** [233] and **FedGLO** [27] since they also propose techniques to improve model accuracy on non-IID datasets. We then compare and evaluate the testing accuracy between our scheme **CLA**, **FedAvg** and **FedGLO**. The class distribution of the parties datasets are imbalanced

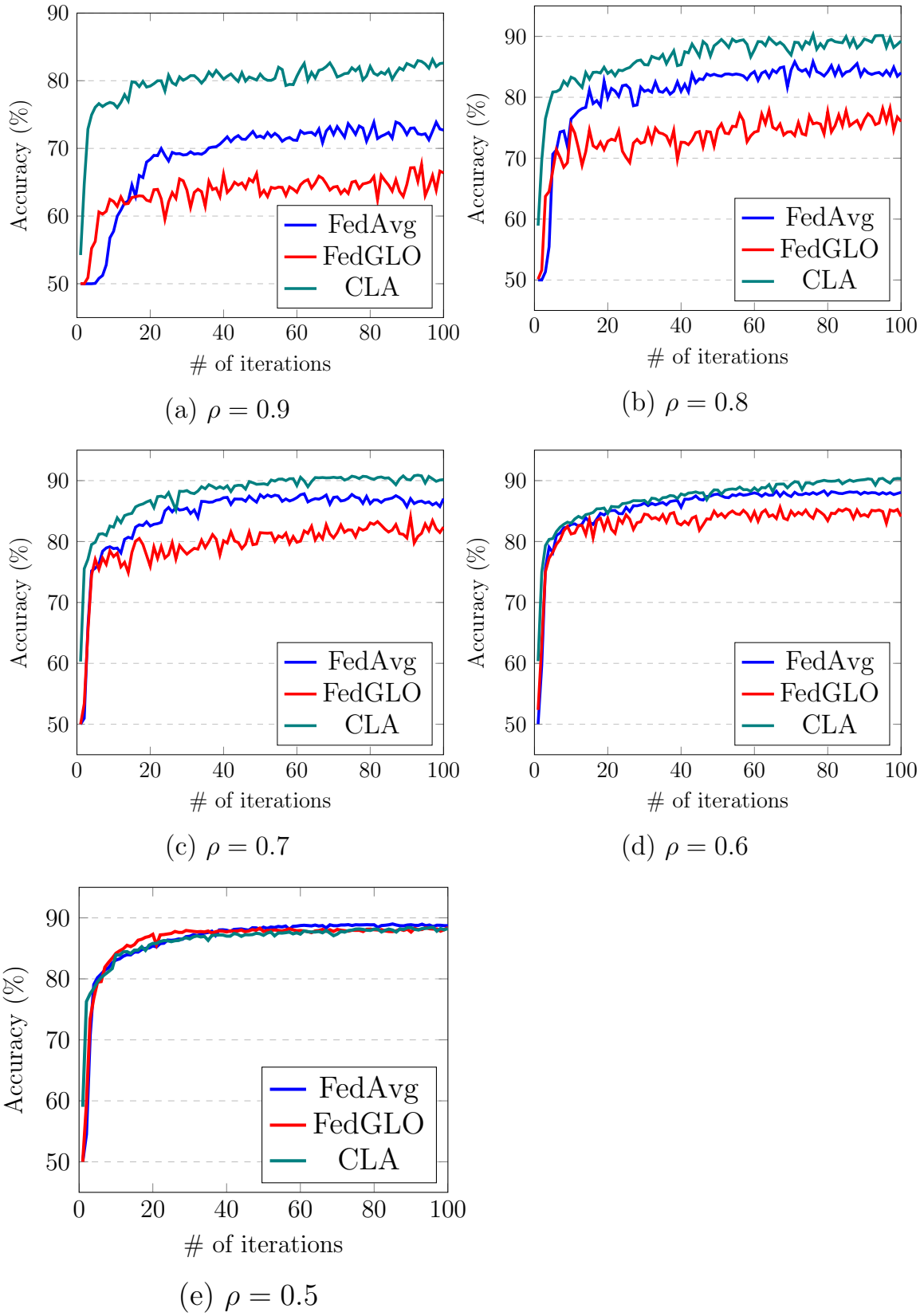


Figure 6.5: Testing accuracy (%) vs. # of federated learning iterations on non-IID training datasets with varying non-IID distribution levels $\rho = 0.9, 0.8, 0.7, 0.6, 0.5$ i.e. the extreme non-IID case ($\rho = 0.9$) to the least non-IID case ($\rho = 0.5$). Our proposed scheme **CLA** outperforms **FedAvg** [233] and **FedGLO** [27] for $\rho = 0.9$ to 0.6.

Table 6.1: Test accuracy comparison of our scheme (CLA) with current state-of-the-art schemes (the bold results are better)

Scheme	**non-IID distribution levels**				
	$\rho = 0.9$	$\rho = 0.8$	$\rho = 0.7$	$\rho = 0.6$	$\rho = 0.5$
FedAvg [233]	72.46%	84.18%	86.85%	87.89%	**88.77%**
FedGLO [27]	64.81%	75.70%	81.75%	84.57%	87.98%
CLA	**81.61%**	**88.93%**	**90.30%**	**89.65%**	87.95%
Improvements					
CLA-FedAvg	9.15%	4.75%	3.45%	1.76%	-0.82%
CLA-FedGLO	16.79%	13.23%	8.55%	5.08%	-0.03%
Notes: **CLA-FedAvg**: Improvement of our scheme CLA over FedAvg [233]; **CLA-FedGLO**: Improvement of our scheme CLA over FedGLO [27]					

(i.e. non-IID) and accuracy will be biased towards the majority class predictor. To address this bias towards high accuracy, we use the balanced accuracy metric to evaluate training accuracy. We collaboratively train the shared model with the three different schemes **CLA**, **FedAvg** and **FedGLO** on heterogeneous genomic datasets with varying non-IID distribution levels $\rho = 0.9, 0.8, 0.7, 0.6, 0.5$, and present the average testing accuracy in percentage (over 10 runs) as depicted in Figure 6.5. It is evident from Figure 6.5 that the accuracy of all schemes increases as the DNA sequence classes becomes less heterogeneous from $\rho = 0.9$ (extreme non-IID case) to $\rho = 0.5$ (least non-IID case) with our scheme **CLA** outperforming **FedAvg** and **FedGLO** for $\rho = 0.9$ to 0.6. It is also evident that our model converges faster than the other techniques. Table 6.1 also presents the accuracies and the improved accuracies of our scheme **CLA** over **FedAvg** and **FedGLO**. These results suggest that our technique counterbalances the bias introduced by the heterogeneous nature of the non-IID local datasets and hence speeds up the model convergence and improves accuracy.

6.6.2 Computation Cost Evaluation

We analyze the computation overhead of our scheme for training and updating the local model (**TUM**), encryption (**ENC**) of gradients on parties, homomorphic gradient aggregation (**HGA**) on the blockchain and threshold decryption (**TDEC**) by parties. We evaluate the computation cost for varying the number of parties from 4 to 10 and present the result of one iteration in Table 6.2. It is evident from Table

6.2 that the computation cost for **TUM** and **ENC** per party is nearly invariable to increasing number of training parties. This is because **TUM** and **ENC** are the local runtime cost on each party. Table 6.2 also shows that homomorphic gradient aggregation slightly increases with an increasing number of training parties. This is because with an increasing number of parties the greater the number of encrypted gradients to aggregate. We also present in Table 6.3 the runtime by varying the number of training iterations from 10 to 100 for $N = 10$. Table 6.3 shows that the computation cost for **TUM** and **ENC** increasing linearly to the number of iterations while the cost of homomorphic gradient aggregation is nearly invariable to increasing iterations. This is because homomorphic gradient aggregation on the blockchain does not depend on the number of global training iterations.

6.6.3 Communication Cost Evaluation

Here we evaluate the total communication cost to upload and download data i.e. encrypted gradients and authentication tags for verification from a learning party to the blockchain. We compute the transmission cost in kilobytes(KB) per party for a varying number of learning iterations and present our results in Figure 6.6. Figure 6.6 shows that the communication cost increases linearly to the number of iterations. The higher communication cost recorded by our scheme is a result of multiple rounds of interaction between parties and the blockchain as the number of iterations increases. Another observation worth noting is that each party transmits approximately 789,492

Table 6.2: Computation cost (seconds) to train the shared model by varying the number of training parties

# of Parties	**TUM**	**ENC**	**HGA**	**TDEC**	**Total**
4	1.1298	0.713	0.1055	0.0437	1.992
5	1.1943	0.7079	0.1168	0.0382	2.0572
6	1.2635	0.7169	0.1382	0.0384	2.157
7	1.3712	0.7428	0.1429	0.0391	2.296
8	1.2636	0.7092	0.1807	0.0397	2.1932
9	1.1494	0.7075	0.2066	0.0395	2.103
10	1.2819	0.8153	0.3117	0.0385	2.4474

Notes: **ENC**: Encryption; **HGA**: Homomorphic gradient aggregation; **TDEC**: Threshold decryption; **TUM**: Training and updating local model

Table 6.3: Computation cost (seconds) to train a shared model by varying the number of iterations

# of Iterations	TUM	ENC	HGA	TDEC	Total
10	13.2194	7.3039	0.345	0.0399	20.9082
20	28.4387	13.6078	0.3478	0.0405	42.4348
30	41.6581	21.9117	0.3987	0.0404	64.0089
40	56.8775	27.2155	0.3265	0.0389	84.4584
50	71.0969	36.5194	0.2985	0.0382	107.953
60	84.3162	44.8234	0.3112	0.0389	129.4897
70	97.5356	51.1272	0.2456	0.0409	148.9493
80	114.7549	57.4311	0.3784	0.0398	172.6042
90	126.9743	66.7349	0.2879	0.0417	194.0388
100	142.1937	73.0388	0.3465	0.0427	215.6217

Notes: **ENC**: Encryption; **HGA**: Homomorphic gradient aggregation; **TDEC**: Threshold decryption; **TUM**: Training and updating local model

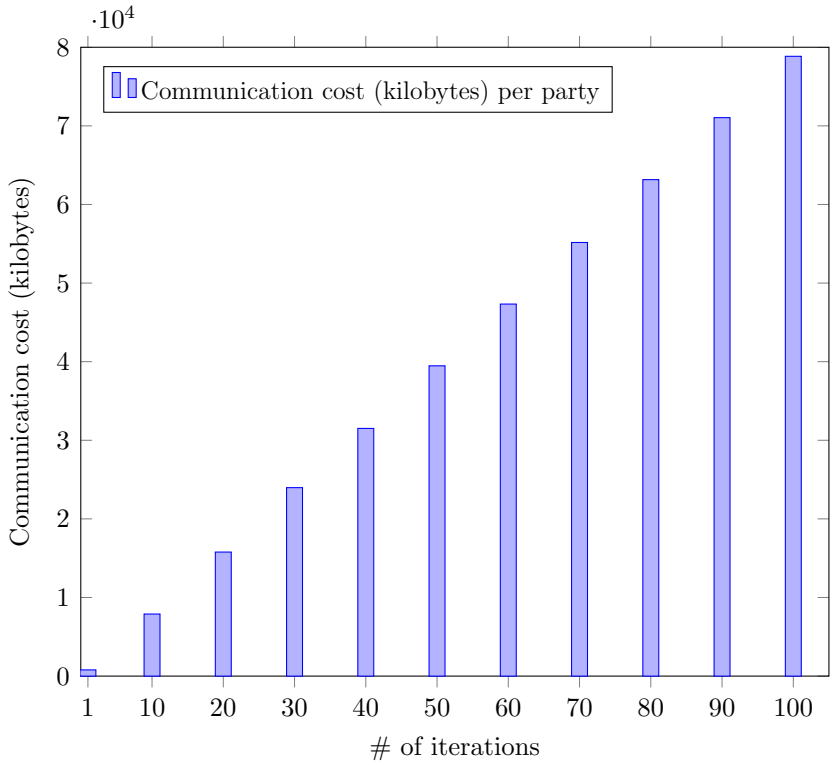


Figure 6.6: Total communication cost (kilobytes) per party to transmit data to and from the blockchain with a varying number of global learning iterations from 1 to 100.

KB within 1 iteration round which is practically feasible.

6.7 Conclusion and Future Work

In this chapter, we proposed a decentralized blockchain FL framework to address the FL challenges of security, verifiability of aggregated gradients and degraded model accuracy for DNA sequence classification across multiple collaborating parties. We employ fully homomorphic encryption to protect the confidentiality of local gradients. We improve FL accuracy by proposing a new technique to create an initialization model trained on random DNA sequences. These DNA sequences are sampled within the Shannon entropy range across parties' DNA sequences. In addition, we develop a new blockchain consensus algorithm using bilinear pairing and homomorphic hash functions to enable the parties to verify the correctness of the aggregated gradients on the blockchain. Finally, we conduct extensive experiments to train a model on the ChIP-seq dataset to predict DNA-binding motifs, and demonstrate that our scheme is practically efficient and yields higher accuracy than current-state-of-art techniques. Our framework results in higher communication rounds and transmission cost. As future work, we will explore communication reduction and compression techniques to reduce transmission overhead. In addition, we will extend our framework to address the challenge of parties dropping out during the FL training process.

Chapter 7: Conclusion and Future Work

In this chapter, we conclude the research conducted in this thesis and discuss the limitations and future directions of this work. We summarize the proposed methodologies and the contributions of this thesis on Section 7.1. In Section 7.2, we discuss the limitations of this research and the potential solutions are provided as future work.

7.1 Conclusion

Genomic privacy is an important multidisciplinary research area which brings researchers in the fields of policymaking, cryptography and biomedical sciences together to enact policies, propose and develop secure genomic storage and computational systems to preserve the privacy of the human genome. Protecting the privacy of individuals' genomic information will encourage them to contribute their genetic samples to repositories. This will drive and advance biomedical research, precision medicine and clinical diagnostics to (i) discover disease causing genetic variants, (ii) discover new drugs, and (iii) provide insights and a better understanding of the structures and functionalities of the human genome. This thesis makes significant contributions to genomic privacy by investigating how to protect the security and privacy of individuals' genomic information at storage and during the computation of genomic analytical tasks with minimal runtime cost and transmission throughout. We studied and developed novel privacy-preserving genomic frameworks using cryptographic approaches such as homomorphic encryption, secure multiparty computation, digital signatures and non-interactive zero knowledge proofs to guarantee confidentiality, privacy, access control and availability from a polynomial time adversary. Specifically, this thesis addresses security and privacy issues in genomic application areas of personalized medicine, genomic access control and variant discovery on the blockchain, and federated deep learning for DNA sequence classification on heterogenous genomic datasets.

In Chapter 2, we examined the gaps in genomic privacy by investigating the current privacy attacks on genomic data and the cryptographic techniques (i.e., homomorphic

encryption, secure cryptographic hardware, differential privacy and secure multiparty computation) that have been employed to protect genomic privacy. In addition, we comprehensively investigated and classified current state-of-the-art genomic privacy-preserving techniques by their genomic application domain, and provided a comparative analysis of these techniques based on their encryption techniques, security goals and computation and transmission overheads. Our literature review highlighted the fact that learning neural network models in personalized medicine applications for the classification disease causing genes (e.g., BRCA1 and BRCA2 gene for breast cancer) while preserving the privacy of patients still remains a challenge.

Our privacy-based drug dosage prediction framework for personalized medicine (Chapter 3) allows the computation of patients' warfarin dosages on encrypted genomic and clinical data. Our novel homomorphic genotype guided drug dosage protocols with secure multiparty computation techniques does not reveal patients' sensitive information. To address the high computation cost incurred by bit decomposition in current state-of-the-art secure comparison protocols, we developed a new SMC protocol using homomorphic and blinding techniques to securely compare patients' single nucleotide polymorphism (SNP) states. In addition, we encode patients' genetic variants, race and age with a new encoding technique to facilitate efficient matching operations during the computation of dosages. The experiment results and security analysis revealed that our proposed scheme is semantically secure and our secure estimated warfarin dosages are similar to that of plaintext computed dosages with negligible losses.

Our proposed blockchain smart contract integration to control genomic access and variant discovery (chapter 4) proposes a new technique to compensate genomic data owners (DOs) with cryptocurrency for contributing their genetic information. Smart contract protocols are developed to enable DOs to seamlessly upload and control access to their data on the blockchain. In addition, a novel genomic variant discovery protocol is developed with homomorphic and secure two-party techniques to enable genomic data users (DUs) run queries to securely discover DOs of interest. Our genomic variant discovery protocol optimized the query response time with novel techniques

based on genomic data partitions, binary search trees and bloom filters to reduce the search space. The experiment results demonstrated that our proposed scheme is effective and efficient in computation cost, query response time and scalability.

The blockchain network is a new technology with the potential to revolutionize how genomic data are collated and accessed for application areas in research and pharmaceuticals. Chapter 5 investigated how blockchain functionalities can be integrated with genomic applications to address the issues of privacy, access control, interoperability and auditability. It further examined and classified blockchain functionality models (i.e., storage, consent and compensation models) with relevant genomic application use cases. In addition, current state-of-the-art blockchain-based genomic and phenotypic data sharing and analytics techniques were investigated and analysed comparatively based on security, blockchain functionalities and transaction complexities. We believe this literature review will serve as a guide for policymakers and researchers to explore how blockchain can be leveraged in genomic applications.

Finally, we proposed secure federated learning for DNA sequence classification (Chapter 6) which protects the privacy of the learning parties' local DNA sequences and model gradients. This scheme enables parties to verify the correctness of aggregated model gradients on the blockchain via a new consensus algorithm (proof of aggregated gradients) using bilinear pairing and homomorphic hash functions. The evaluation results demonstrate that our approach is efficient and yields superior model accuracy compared to current state-of-the-art techniques.

7.2 Future Directions and Open Problems

Personalized medicine is still growing and researchers are actively working to develop machine learning pharmacogenetic models capable of predicting drug dosages for various health conditions. In Chapter 3, this thesis focused on securing the most popular regression model for predicting warfarin drug dosage for the treatment of thromboembolic disorders (i.e., treatment of blood clots that might cause stroke or heart conditions etc.). Our proposed scheme only focuses on protecting the privacy of patients undergoing treatment for thromboembolic disorders. As future work, our proposed

framework can be extended to secure patients' genomic and clinical information used by pharmacogenetic models to predict Tamoxifen for the treatment of breast cancer. Our proposed secure frameworks for dosage prediction in Chapter 3 and blockchain-based genomic access control and variant discovery in Chapter 4 address a security model where parties are honest-but-curious. This means that parties faithfully follow the protocol but might try to learn an individual's sensitive genomic information. The limitation of the honest-but-curious security model is that it does not address malicious behaviors where a party can tamper with genomic computation by injecting false data or returning false computation results for reasons of personal gain. We propose as future work that our proposed cryptographic techniques in Chapters 3 and Chapter 4 can be extended to utilize zero knowledge proof (either interactive or non-interactive zero knowledge proof) to allow computing parties to prove to a verifier that they have indeed performed the correct computation or returned the right results.

Finally, our proposed secure federated learning for DNA sequence classification framework (Chapter 6) assumes that learning parties within the federated learning collaborative group do not drop out of the training process. However, it is possible for some parties to drop out due to unreliable network issues and as a result their model gradients will not be aggregated to the shared model for some training rounds. As future work, we will investigate how to extend our framework with cryptographic homomorphic subtraction techniques to tolerate parties dropping out during the training process. In addition, our secure federated learning framework incurs higher transmission costs as a result of several communication rounds as reported in the experiment result in Chapter 6. As future work, we will explore ciphertext compression techniques to reduce the transmission payload.

References

[1] C. Bycroft, C. Freeman, D. Petkova, G. Band, *et al.*, "The uk biobank resource with deep phenotyping and genomic data," *Nature*, vol. 562, no. 7726, pp. 203–209, 2018.

[2] C. A. Davis, B. C. Hitz, C. A. Sloan, E. T. Chan, *et al.*, "The encyclopedia of dna elements (encode): data portal update," *Nucleic acids research*, vol. 46, no. D1, pp. D794–D801, 2018.

[3] X. Zheng-Bradley and P. Flicek, "Applications of the 1000 genomes project resources," *Briefings in functional genomics*, vol. 16, no. 3, pp. 163–170, 2017.

[4] M. Claussnitzer, J. H. Cho, R. Collins, N. J. Cox, *et al.*, "A brief history of human disease genetics," *Nature*, vol. 577, no. 7789, pp. 179–189, 2020.

[5] X. Li, D. Li, J.-C. Wu, Z.-Q. Liu, *et al.*, "Precision dosing of warfarin: open questions and strategies," *The pharmacogenomics journal*, p. 1, 2019.

[6] H. Cho, D. J. Wu, and B. Berger, "Secure genome-wide association analysis using multiparty computation," *Nature biotechnology*, vol. 36, no. 6, p. 547, 2018.

[7] M. Naveed, E. Ayday, E. W. Clayton, J. Fellay, *et al.*, "Privacy in the genomic era," *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, p. 6, 2015.

[8] G. Cloud, "Cloud life sciences," 2021. https://cloud.google.com/life-sciences.

[9] Microsoft, "Microsoft genomics," 2021. https://azure.microsoft.com/en-us/services/genomics/.

[10] M. Shabani and P. Borry, "Rules for processing genetic data for research purposes in view of the new eu general data protection regulation," *European Journal of Human Genetics*, vol. 26, no. 2, pp. 149–156, 2018.

[11] E. W. Clayton, B. J. Evans, J. W. Hazel, and M. A. Rothstein, "The law of genetic privacy: applications, implications, and limitations," *Journal of Law and the Biosciences*, vol. 6, no. 1, pp. 1–36, 2019.

[12] M. Humbert, E. Ayday, J.-P. Hubaux, and A. Telenti, "Quantifying interdependent risks in genomic privacy," *ACM Transactions on Privacy and Security (TOPS)*, vol. 20, no. 1, p. 3, 2017.

[13] Y. Erlich, T. Shor, I. Pe'er, and S. Carmi, "Identity inference of genomic data using long-range familial searches," *Science*, vol. 362, no. 6415, pp. 690–694, 2018.

[14] Z. He, J. Yu, J. Li, Q. Han, *et al.*, "Inference attacks and controls on genotypes and phenotypes for individual genomic data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, no. 1, pp. 1–1, 2018.

[15] Y. Lindell, *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*. Springer, 2017.

[16] R. Zhang, R. Xue, and L. Liu, "Security and privacy on blockchain," *ACM Computing Surveys (CSUR)*, vol. 52, no. 3, pp. 1–34, 2019.

[17] Y. Erlich and A. Narayanan, "Routes for breaching and protecting genetic privacy," *Nature Reviews Genetics*, vol. 15, no. 6, pp. 409–421, 2014.

[18] M. M. A. Aziz, M. N. Sadat, D. Alhadidi, S. Wang, *et al.*, "Privacy-preserving techniques of genomic data-a survey," *Briefings in bioinformatics*, 2017.

[19] S. Wang, X. Jiang, S. Singh, R. Marmor, *et al.*, "Genome privacy: challenges, technical approaches to mitigate risk, and ethical considerations in the united states," *Annals of the New York Academy of Sciences*, vol. 1387, no. 1, pp. 73–83, 2017.

[20] A. Mittos, B. Malin, and E. De Cristofaro, "Systematizing genome privacy research: A privacy-enhancing technologies perspective," *Proceedings on Privacy Enhancing Technologies*, vol. 1, pp. 87–107, 2019.

[21] BBC, "Who's making money from your dna?," 2019. https://www.bbc.com/worklife/article/20190301-how-screening-companies-are-monetising-your-dna.

[22] CNBC, "Gsk strikes $300 million deal with 23andme for genetics-driven drug research," 2018. https://www.cnbc.com/2018/07/24/glaxosmithkline-23andme-team-up-on-genetics-driven-drug-research.html.

[23] F. Briscoe, I. Ajunwa, A. Gaddis, and J. McCormick, "Evolving public views on the value of one's dna and expectations for genomic database governance: Results from a national survey," *PloS one*, vol. 15, no. 3, p. e0229044, 2020.

[24] H. I. Ozercan, A. M. Ileri, E. Ayday, and C. Alkan, "Realizing the potential of blockchain technologies in genomics," *Genome research*, vol. 28, no. 9, pp. 1255–1263, 2018.

[25] L. DeFrancesco and A. Klevecz, "Your dna broker," *Nature biotechnology*, vol. 37, no. 8, pp. 842–847, 2019.

[26] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–36, 2021.

[27] Y. Zhao, M. Li, L. Lai, N. Suda, *et al.*, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[28] W. Chen, K. Bhardwaj, and R. Marculescu, "Fedmax: mitigating activation divergence for accurate and communication-efficient federated learning," *arXiv preprint arXiv:2004.03657*, 2020.

[29] Australian Genomics. A National Approach to Data Federation and Analysis, 2021. [Online]. Available: https://www.australiangenomics.org.au.

[30] STAT. House Republicans would let employers demand worker's genetic results, 2017. [Online]. Available: https://www.statnews.com/2017/03/10/workplace-wellness-genetic-testing.

[31] Daily Mail. DNA test for every baby, 2013. [Online]. Available: https://www.dailymail.co.uk/health/article-186118/DNA-test-baby.html.

[32] NIH, "Genetic discrimination and other laws," 2017. https://www.genome.gov/27568503/genetic-discrimination-and-other-laws/.

[33] CNBC. Risks of sharing your DNA, 2018. [Online]. Available: https://www.cnbc.com/2018/06/16/5-biggest-risks-of-sharing-dna-with-consumer-genetic-testing-companies.html.

[34] EU, "General data protection regulation," *Official Journal of the European Union*, vol. L119, pp. 1–88, 2016.

[35] M. Humbert, E. Ayday, J.-P. Hubaux, and A. Telenti, "Addressing the concerns of the lacks family: quantification of kin genomic privacy," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 1141–1152, ACM, 2013.

[36] G. Kale, E. Ayday, and O. Tastan, "A utility maximizing and privacy preserving approach for protecting kinship in genomic databases," *Bioinformatics*, vol. 34, no. 2, pp. 181–189, 2017.

[37] M. Akgün, A. O. Bayrak, B. Ozer, and M. Ş. Sağıroğlu, "Privacy preserving processing of genomic data: A survey," *Journal of biomedical informatics*, vol. 56, pp. 103–111, 2015.

[38] L. Sweeney, A. Abu, and J. Winn, "Identifying participants in the personal genome project by name (a re-identification experiment)," *Harvard University. Data Privacy Lab. White Paper 1021-1*, 2013.

[39] M. Gymrek, A. L. McGuire, D. Golan, E. Halperin, *et al.*, "Identifying personal genomes by surname inference," *Science*, vol. 339, no. 6117, pp. 321–324, 2013.

[40] S. S. Shringarpure and C. D. Bustamante, "Privacy risks from genomic data-sharing beacons," *The American Journal of Human Genetics*, vol. 97, no. 5, pp. 631–646, 2015.

[41] J. L. Raisaro, F. Tramèr, Z. Ji, D. Bu, *et al.*, "Addressing beacon re-identification attacks: quantification and mitigation of privacy risks," *Journal of the American Medical Informatics Association*, vol. 24, no. 4, pp. 799–805, 2017.

[42] N. von Thenen, E. Ayday, and A. E. Cicek, "Re-identification of individuals in genomic data-sharing beacons via allele inference," *Bioinformatics*, p. bty643, 2018.

[43] N. Homer, S. Szelinger, M. Redman, D. Duggan, *et al.*, "Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays," *PLoS genetics*, vol. 4, no. 8, p. e1000167, 2008.

[44] M. Fredrikson, E. Lantz, S. Jha, S. Lin, *et al.*, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing.," in *USENIX Security Symposium*, pp. 17–32, 2014.

[45] M. Humbert, K. Huguenin, J. Hugonot, E. Ayday, and J.-P. Hubaux, "De-anonymizing genomic databases using phenotypic traits," *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 99–114, 2015.

[46] R. Cai, Z. Hao, M. Winslett, X. Xiao, *et al.*, "Deterministic identification of specific individuals from gwas results," *Bioinformatics*, vol. 31, no. 11, pp. 1701–1707, 2015.

[47] C. Lippert, R. Sabatini, M. C. Maher, E. Y. Kang, *et al.*, "Identification of individuals by trait prediction using whole-genome sequencing data," *Proceedings of the National Academy of Sciences*, vol. 114, no. 38, pp. 10166–10171, 2017.

[48] A. Kong, G. Masson, M. L. Frigge, A. Gylfason, *et al.*, "Detection of sharing by descent, long-range phasing and haplotype imputation," *Nature genetics*, vol. 40, no. 9, p. 1068, 2008.

[49] I. Deznabi, M. Mobayen, N. Jafari, O. Tastan, *et al.*, "An inference attack on genomic data using kinship, complex correlations, and phenotype information," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2017.

[50] GA4GH. The Beacon project, 2018. [Online]. Available: https://beacon-network.org.

[51] R. Wang, Y. F. Li, X. Wang, H. Tang, *et al.*, "Learning your identity and disease from research papers: information leaks in genome wide association study," in *Proceedings of the 16th ACM conference on Computer and communications security*, pp. 534–544, ACM, 2009.

[52] Y. Erlich, "Major flaws in "identification of individuals by trait prediction using whole-genome"."

[53] J. Marchini and B. Howie, "Genotype imputation for genome-wide association studies," *Nature Reviews Genetics*, vol. 11, no. 7, p. 499, 2010.

[54] A. Johnson and V. Shmatikov, "Privacy-preserving data exploration in genome-wide association studies," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1079–1087, ACM, 2013.

[55] E. A. Zerhouni and E. G. Nabel, "Protecting aggregate genomic data," *Science*, vol. 322, no. 5898, pp. 44–44, 2008.

[56] C. Dwork, A. Roth, *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[57] A. K. Daly, "Genome-wide association studies in pharmacogenomics," *Nature Reviews Genetics*, vol. 11, no. 4, p. 241, 2010.

[58] S. Simmons and B. Berger, "Realizing privacy preserving genome-wide association studies," *Bioinformatics*, vol. 32, no. 9, pp. 1293–1300, 2016.

[59] Z. Huang, H. Lin, J. Fellay, Z. Kutalik, *et al.*, "Sqc: secure quality control for meta-analysis of genome-wide association studies," *Bioinformatics*, vol. 33, no. 15, pp. 2273–2280, 2017.

[60] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Comp. Surv.*, vol. 51, no. 4, pp. 1–35, 2018.

[61] P. Laud and L. Kamm, *Applications of Secure Multiparty Computation*, vol. 13. Ios Press, 2015.

[62] IBM. IBM cryptographic coprocessor, 2017. [Online]. Available: http://www.ibm.com/security.

[63] A. C. Yao, "Protocols for secure computations," in *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pp. 160–164, IEEE, 1982.

[64] R. Wang, X. Wang, Z. Li, H. Tang, *et al.*, "Privacy-preserving genomic computation through program specialization," in *Proceedings of the 16th ACM conference on Computer and communications security*, pp. 338–347, ACM, 2009.

[65] M. Blanton and F. Bayatbabolghani, "Improving the security and efficiency of private genomic computation using server aid," *IEEE Security & Privacy*, no. 5, pp. 20–28, 2017.

[66] F. Bruekers, S. Katzenbeisser, K. Kursawe, and P. Tuyls, "Privacy-preserving matching of dna profiles.," *IACR Cryptology ePrint Archive*, vol. 2008, p. 203, 2008.

[67] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, *et al.*, "Innovative instructions and software model for isolated execution.," *HASP@ ISCA*, vol. 10, 2013.

[68] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for cpu based attestation and sealing," in *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, vol. 13, ACM New York, NY, USA, 2013.

[69] ARM. TrustZone Technology for Microcontrollers, 2018. [Online]. Available: https://www.arm.com/why-arm/technologies/trustzone-for-cortex-m.

[70] S. Trimberger and J. Moore, "Fpga security: From features to capabilities to trusted systems," in *Proceedings of the 51st Annual Design Automation Conference*, pp. 1–4, ACM, 2014.

[71] L. Xu, H. Kim, X. Wang, W. Shi, and T. Suh, "Privacy preserving large scale dna read-mapping in mapreduce framework using fpgas," in *Field Programmable Logic and Applications (FPL), 2014 24th International Conference on*, pp. 1–4, IEEE, 2014.

[72] F. Chen, S. Wang, X. Jiang, S. Ding, *et al.*, "Princess: Privacy-protecting rare disease international network collaboration via encryption through software guard extensions," *Bioinformatics*, vol. 33, no. 6, pp. 871–878, 2016.

[73] F. Chen, C. Wang, W. Dai, X. Jiang, *et al.*, "Presage: Privacy-preserving genetic testing via software guard extension," *BMC medical genomics*, vol. 10, no. 2, p. 48, 2017.

[74] M. N. Sadat, M. M. Al Aziz, N. Mohammed, F. Chen, *et al.*, "Safety: secure gwas in federated environment through a hybrid solution," *IEEE/ACM trans. on comput. biology and bioinfo.*, vol. 16, no. 1, pp. 93–102, 2018.

[75] J. R. Troncoso-Pastoriza and F. Perez-Gonzalez, "Secure signal processing in the cloud: enabling technologies for privacy-preserving multimedia cloud processing," *IEEE Signal Processing Magazine*, vol. 30, no. 2, pp. 29–41, 2013.

[76] K. Ligett, S. Neel, A. Roth, B. Waggoner, *et al.*, "Accuracy first: Selecting a differential privacy level for accuracy constrained erm," in *Advances in Neural Information Processing Systems*, pp. 2566–2576, 2017.

[77] H. Tang, X. Jiang, X. Wang, S. Wang, *et al.*, "Protecting genomic data analytics in the cloud: state of the art and opportunities," *BMC Med. Geno.*, vol. 9, no. 1, p. 63, 2016.

[78] S. Wang, X. Jiang, H. Tang, X. Wang, *et al.*, "A community effort to protect genomic data sharing, collaboration and outsourcing," *NPJ genomic medicine*, vol. 2, no. 1, p. 33, 2017.

[79] K. A. Jagadeesh, D. J. Wu, J. A. Birgmeier, D. Boneh, *et al.*, "Deriving genomic diagnoses without revealing patient genomes," *Science*, vol. 357, no. 6352, pp. 692–695, 2017.

[80] Y. Lindell and J. Katz, *Introduction to modern cryptography*. Chapman and Hall/CRC, 2014.

[81] M. Kantarcioglu, W. Jiang, Y. Liu, and B. Malin, "A cryptographic approach to securely share and query genomic sequences," *IEEE Transactions on information technology in biomedicine*, vol. 12, no. 5, pp. 606–617, 2008.

[82] M. Canim, M. Kantarcioglu, and B. Malin, "Secure management of biomedical data with cryptographic hardware," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 1, pp. 166–175, 2012.

[83] R. Ghasemi, M. M. Al Aziz, N. Mohammed, M. H. Dehkordi, *et al.*, "Private and efficient query processing on outsourced genomic databases," *IEEE journal of biomedical and health informatics*, vol. 21, no. 5, pp. 1466–1472, 2017.

[84] M. Nassar, Q. Malluhi, M. Atallah, and A. Shikfa, "Securing aggregate queries for dna databases," *IEEE Transactions on Cloud Computing*, no. 1, pp. 1–1, 2017.

[85] M. Z. Hasan, M. S. R. Mahdi, M. N. Sadat, and N. Mohammed, "Secure count query on encrypted genomic data," *Journal of biomedical informatics*, vol. 81, pp. 41–52, 2018.

[86] F. Tramèr, Z. Huang, J.-P. Hubaux, and E. Ayday, "Differential privacy with bounded priors: reconciling utility and privacy in genome-wide association studies," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1286–1297, ACM, 2015.

[87] L. Kamm, D. Bogdanov, S. Laur, and J. Vilo, "A new way to protect privacy in large-scale genome-wide association studies," *Bioinformatics*, vol. 29, no. 7, pp. 886–893, 2013.

[88] B. Dan, Liina, L. Swen, and S. Ville, "Implementation and evaluation of an algorithm for cryptographically private principal component analysis on genomic data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2018.

[89] M. J. Atallah and J. Li, "Secure outsourcing of sequence comparisons," *International Journal of Information Security*, vol. 4, no. 4, pp. 277–287, 2005.

[90] S. Jha, L. Kruger, and V. Shmatikov, "Towards practical privacy for genomic computation," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pp. 216–230, IEEE, 2008.

[91] G. Asharov, S. Halevi, Y. Lindell, and T. Rabin, "Privacy-preserving search of similar patients in genomic data.," *IACR Cryptology ePrint Archive*, vol. 2017, p. 144, 2017.

[92] M. M. Al Aziz, D. Alhadidi, and N. Mohammed, "Secure approximation of edit distance on genomic data," *BMC medical genomics*, vol. 10, no. 2, p. 41, 2017.

[93] M. S. R. Mahdi, M. M. Al Aziz, D. Alhadidi, and N. Mohammed, "Secure similar patients query on encrypted genomic data," *IEEE journal of bio. and health infor.*, vol. 23, no. 6, pp. 2611–2618, 2018.

[94] J. S. Sousa, C. Lefebvre, Z. Huang, J. L. Raisaro, *et al.*, "Efficient and secure outsourcing of genomic data storage," *BMC medical genomics*, vol. 10, no. 2, p. 46, 2017.

[95] B. Wang, W. Song, W. Lou, and Y. T. Hou, "Privacy-preserving pattern matching over encrypted genetic data in cloud computing," in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, pp. 1–9, IEEE, 2017.

[96] X. Wang and Y. Zhang, "E-sc: Collusion-resistant secure outsourcing of sequence comparison algorithm," *IEEE ACCESS*, vol. 6, pp. 3358–3375, 2018.

[97] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik, "Privacy preserving error resilient dna searching through oblivious automata," in *Proceedings of the*

14th ACM conference on Computer and communications security, pp. 519–528, ACM, 2007.

[98] P. J. McLaren, J. L. Raisaro, M. Aouri, M. Rotger, *et al.*, "Privacy-preserving genomic testing in the clinic: a model using hiv treatment," *Genetics in medicine*, vol. 18, no. 8, p. 814, 2016.

[99] M. M. Al Aziz, R. Ghasemi, M. Waliullah, and N. Mohammed, "Aftermath of bustamante attack on genomic beacon service," *BMC medical genomics*, vol. 10, no. 2, p. 43, 2017.

[100] Z. Wan, Y. Vorobeychik, M. Kantarcioglu, and B. Malin, "Controlling the signal: Practical privacy protection of genomic data sharing through beacon services," *BMC medical genomics*, vol. 10, no. 2, p. 39, 2017.

[101] Deep Genomics. Creating A New Universe Of Genetic Medicines, 2018. [Online]. Available: https://www.deepgenomics.com.

[102] Atomwise. Artificial Intelligence for Drug Discovery, 2018. [Online]. Available: https://www.atomwise.com.

[103] S. Delhalle, S. F. Bode, R. Balling, M. Ollert, and F. Q. He, "A roadmap towards personalized immunology," *NPJ systems biology and applications*, vol. 4, no. 1, p. 9, 2018.

[104] C. Y. Lee and Y.-P. P. Chen, "Machine learning on adverse drug reactions for pharmacovigilance," *Drug discovery today*, 2019.

[105] F. Carrasco-Ramiro, R. Peiro-Pastor, and B. Aguado, "Human genomics projects and precision medicine," *Gene therapy*, vol. 24, no. 9, p. 551, 2017.

[106] M. Ozturk, A. Ipekci, S. K. Kiyak, Y. S. Akdeniz, *et al.*, "Bleeding complications in warfarin-treated patients admitted to the emergency department," *Journal of clinical medicine research*, vol. 11, no. 2, p. 106, 2019.

[107] USFDA, "Nucleic acid based tests," 2021. https://www.fda.gov/medical-devices/in-vitro-diagnostics/nucleic-acid-based-tests.

[108] J. A. Johnson *et al.*, "Clinical pharmacogenetics implementation consortium (cpic) guideline for pharmacogenetics-guided warfarin dosing: 2017 update," *Clin. Phar. & Thera.*, vol. 102, no. 3, pp. 397–404, 2017.

[109] A. Yakubu and Y.-P. Chen, "Ensuring privacy and security of genomic data and functionalities," *Brief. in Bioinfo.*, vol. 21, no. 2, pp. 511–526, 2020.

[110] J. Wei *et al.*, "Differential privacy-based genetic matching in personalized medicine," *IEEE Trans. on Emerging Topics in Computing*, 2020.

[111] F. Chen, S. Wang, X. Jiang, S. Ding, *et al.*, "Princess: Privacy-protecting rare disease international network collaboration via encryption through software guard extensions," *Bioinformatics*, vol. 33, no. 6, pp. 871–878, 2017.

[112] A. Wood, K. Najarian, and D. Kahrobaei, "Homomorphic encryption for machine learning in medicine and bioinformatics," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–35, 2020.

[113] X. Liu, R. Deng, K.-K. R. Choo, Y. Yang, *et al.*, "Privacy-preserving outsourced calculation toolkit in the cloud," *IEEE Trans. on Dependable and Secure Computing*, 2018.

[114] P. Avery, A. Jorgensen, A.-K. Hamberg, M. Wadelius, M. Pirmohamed, F. Kamali, and E.-P. S. Group, "A proposal for an individualized pharmacogenetics-based warfarin initiation dose regimen for patients commencing anticoagulation therapy," *Clinical Pharmacology & Therapeutics*, vol. 90, no. 5, pp. 701–706, 2011.

[115] IWPC, "Warfarin dosing," 2016. http://www.warfarindosing.org.

[116] M. Nateghizad, T. Veugen, Z. Erkin, and R. L. Lagendijk, "Secure equality testing protocols in the two-party setting," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, p. 3, ACM, 2018.

[117] R. Rajasekaran and Y.-P. P. Chen, "Potential therapeutic targets and the role

of technology in developing novel antileishmanial drugs," *Drug discovery today*, vol. 20, no. 8, pp. 958–968, 2015.

[118] E. Uffelmann, Q. Q. Huang, N. S. Munung, J. de Vries, *et al.*, "Genome-wide association studies," *Nature Reviews Methods Primers*, vol. 1, no. 1, pp. 1–21, 2021.

[119] A. Harmanci and M. Gerstein, "Quantification of private information leakage from phenotype-genotype data: linking attacks," *Nature methods*, vol. 13, no. 3, pp. 251–256, 2016.

[120] PharmGKB, "Vkorc1 allele definition table," 2017. https://www.pharmgkb.org/page/vkorc1RefMaterials.

[121] T. Ruzickova, M. Sramek, V. Kaplan, S. Kumstyrova, *et al.*, "Warfarin loading dose guided by pharmacogenetics is effective and safe in cardioembolic stroke patients–a randomized, prospective study," *The pharmacogenomics journal*, p. 1, 2019.

[122] Q. Li, H. Tao, J. Wang, Q. Zhou, *et al.*, "Warfarin maintenance dose prediction for patients undergoing heart valve replacement—a hybrid model with genetic algorithm and back-propagation neural network," *Scientific reports*, vol. 8, no. 1, p. 9712, 2018.

[123] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1–30, 2006.

[124] R. Cramer, I. Damgard, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[125] Z. Shan, K. Ren, M. Blanton, and C. Wang, "Practical secure computation outsourcing: a survey," *ACM Comp. Surv.*, vol. 51, no. 2, p. 31, 2018.

[126] R. Ma, Y. Li, C. Li, F. Wan, *et al.*, "Secure multiparty computation for privacy-preserving drug discovery," *Bioinformatics*, vol. 36, no. 9, pp. 2872–2880, 2020.

[127] Y. Zheng, H. Cui, C. Wang, and J. Zhou, "Privacy-preserving image denoising from external cloud databases," *IEEE Trans. on Infor. Forensics and Security*, vol. 12, no. 6, pp. 1285–1298, 2017.

[128] U. M. Ascher and C. Greif, *A first course on numerical methods*, vol. 7. Siam, 2011.

[129] S. G. Teo, J. Cao, and V. C. Lee, "Dag: A general model for privacy-preserving data mining," *IEEE Trans. on Knowledge and Data Engineering*, vol. 32, no. 1, pp. 40–53, 2018.

[130] PharmGKB, "Pharmgkb dataset downloads," 2018. https://www.pharmgkb.org/downloads.

[131] HealthIT, "Recommended bandwidth for different types of health care providers," 2019. https://www.healthit.gov/faq/what-recommended-bandwidth-different-types-health-care-providers.

[132] U. Health, "Warfarin tablet identification," 2019. https://health.ucsd.edu/specialties/anticoagulation/providers/warfarin/Pages/tablet-identification.aspx.

[133] E. Barker and Q. Dang, "Nist special publication 800-57 part 1, revision 4," *NIST, Tech. Rep*, 2016.

[134] J. L. Raisaro, G. Choi, S. Pradervand, R. Colsenet, *et al.*, "Protecting privacy and security of genomic data in i2b2 with homomorphic encryption and differential privacy," *IEEE/ACM trans. on comp. biology and bioinfo.*, vol. 15, no. 5, pp. 1413–1426, 2018.

[135] O. Chervova, L. Conde, J. A. Guerra-Assunção, I. Moghul, *et al.*, "The personal genome project-uk, an open access resource of human multi-omics data," *Scientific data*, vol. 6, no. 1, pp. 1–10, 2019.

[136] D. Grishin, K. Obbad, and G. M. Church, "Data privacy in the age of personal genomics," *Nat. biotech.*, vol. 37, no. 10, pp. 1115–1117, 2019.

[137] E. J. De Aguiar, B. S. Faiçal, B. Krishnamachari, and J. Ueyama, "A survey of blockchain-based strategies for healthcare," *ACM Comp. Surv.*, vol. 53, no. 2, pp. 1–27, 2020.

[138] K. Fan, S. Wang, Y. Ren, H. Li, and Y. Yang, "Medblock: Efficient and secure medical data sharing via blockchain," *Journal of medical systems*, vol. 42, no. 8, pp. 1–11, 2018.

[139] J. Sun, X. Yao, S. Wang, and Y. Wu, "Blockchain-based secure storage and access scheme for electronic medical records in ipfs," *IEEE Access*, vol. 8, pp. 59389–59401, 2020.

[140] V. Jaiman and V. Urovi, "A consent model for blockchain-based health data sharing platforms," *IEEE Access*, vol. 8, pp. 143734–143745, 2020.

[141] P. Zhang, J. White, D. C. Schmidt, G. Lenz, and S. T. Rosenbloom, "Fhirchain: applying blockchain to securely and scalably share clinical data," *Computational and structural biotechnology journal*, vol. 16, pp. 267–278, 2018.

[142] R. H. Hylock and X. Zeng, "A blockchain framework for patient-centered health records and exchange (healthchain): evaluation and proof-of-concept study," *Journal of medical Internet research*, vol. 21, no. 8, p. e13592, 2019.

[143] N. Kulemin, S. Popov, and A. Gorbachev, "The zenome project: Whitepaper blockchain-based genomic ecosystem," *Zenome. io*, 2017.

[144] D. Grishin, K. Obbad, P. Estep, M. Cifric, *et al.*, "Nebula genomics: Blockchain-enabled genomic data sharing and analysis platform," *Nebula Genomics.(22 June 2018, https://nebulas.io/docs/NebulasTechnicalWhitepaper.pdf)*.

[145] D. Grishin, J. L. Raisaro, J. R. Troncoso-Pastoriza, K. Obbad, *et al.*, "Citizen-centered, auditable and privacy-preserving population genomics," *Nature Computational Science*, vol. 1, no. 3, pp. 192–198, 2021.

[146] VCFtools, "The variant call format specification vcfv4.3 and bcfv2.2," 2021. http://samtools.github.io/hts-specs/.

[147] R. Chikhi, J. Holub, and P. Medvedev, "Data structures to represent a set of k-long dna sequences," *ACM Comp. Surv.*, vol. 54, no. 1, pp. 1–22, 2021.

[148] NCBI, "The genome reference consortium," 2019. https://www.ncbi.nlm.nih.gov/grc/human.

[149] WHO, "Icd-10: International statistical classification of diseases and related health problems 10th revision," 2019. https://icd.who.int/browse10/2019/en.

[150] M. Belenkiy, M. Chase, C. C. Erway, J. Jannotti, *et al.*, "Incentivizing outsourced computation," in *Proceedings of the 3rd international workshop on Economics of networked systems*, pp. 85–90, 2008.

[151] M. Fiume, M. Cupak, S. Keenan, J. Rambla, *et al.*, "Federated discovery and sharing of genomic data using beacons," *Nature biotechnology*, vol. 37, no. 3, pp. 220–224, 2019.

[152] H. Chen, K. Laine, and R. Player, "Simple encrypted arithmetic library-seal v2. 1," in *International Conference on Financial Cryptography and Data Security*, pp. 3–18, Springer, 2017.

[153] T. magazine, "A major drug company now has access to 23andme's genetic data. should you be concerned?," 2018. https://time.com/5349896/23andme-glaxo-smith-kline/.

[154] M. Gaynor, J. Tuttle-Newhall, J. Parker, A. Patel, *et al.*, "Adoption of blockchain in health care," *Journal of medical Internet research*, vol. 22, no. 9, p. e17423, 2020.

[155] S. E. Chang and Y. Chen, "Blockchain in health care innovation: Literature review and case study from a business ecosystem perspective," *Journal of Medical Internet Research*, vol. 22, no. 8, p. e19480, 2020.

[156] A. Hasselgren, K. Kralevska, D. Gligoroski, S. A. Pedersen, *et al.*, "Blockchain in healthcare and health sciences—a scoping review," *International Journal of Medical Informatics*, vol. 134, p. 104040, 2020.

[157] T.-T. Kuo, H. Zavaleta Rojas, and L. Ohno-Machado, "Comparison of blockchain platforms: a systematic review and healthcare examples," *Journal of the American Medical Informatics Association*, vol. 26, no. 5, pp. 462–478, 2019.

[158] T.-T. Kuo, H.-E. Kim, and L. Ohno-Machado, "Blockchain distributed ledger technologies for biomedical and health care applications," *Journal of the American Medical Informatics Association*, vol. 24, no. 6, pp. 1211–1220, 2017.

[159] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," tech. rep., Manubot, 2019.

[160] A.-S. Kleinaki, P. Mytis-Gkometh, G. Drosatos, P. S. Efraimidis, *et al.*, "A blockchain-based notarization service for biomedical knowledge retrieval," *Computational and structural biotechnology journal*, vol. 16, pp. 288–297, 2018.

[161] Y. Zhuang, L. R. Sheets, Y.-W. Chen, Z.-Y. Shae, *et al.*, "A patient-centric health information exchange framework using blockchain technology," *IEEE journal of biomedical and health informatics*, vol. 24, no. 8, pp. 2169–2176, 2020.

[162] S. Wang, J. Wang, X. Wang, T. Qiu, *et al.*, "Blockchain-powered parallel healthcare systems based on the acp approach," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 4, pp. 942–950, 2018.

[163] V. Patel, "A framework for secure and decentralized sharing of medical imaging data via blockchain consensus," *Health informatics journal*, vol. 25, no. 4, pp. 1398–1411, 2019.

[164] N. D. Pattengale and C. M. Hudson, "Decentralized genomics audit logging via permissioned blockchain ledgering," *BMC Medical Genomics*, vol. 13, no. 7, pp. 1–9, 2020.

[165] M. S. Ozdayi, M. Kantarcioglu, and B. Malin, "Leveraging blockchain for immutable logging and querying across multiple sites," *BMC Medical Genomics*, vol. 13, no. 7, pp. 1–7, 2020.

[166] Bitcoin, "Bitcoin network," 2021. https://bitcoin.org/en/.

[167] Ethereum, "Ethereum blockchain network," 2021. https://ethereum.org/en/.

[168] T. Blummer, M. Sean, and C. Cachin, "An introduction to hyperledger," *Hyperledger Under Linux Found., White Paper*, 2018.

[169] ConsenSys, "Quorum blockchain," 2021. https://consensys.net/quorum/.

[170] A. Yazdinejad, G. Srivastava, R. M. Parizi, A. Dehghantanha, *et al.*, "Decentralized authentication of distributed patients in hospital networks using blockchain," *IEEE journal of biomedical and health informatics*, vol. 24, no. 8, pp. 2146–2156, 2020.

[171] P. Li, C. Xu, H. Jin, C. Hu, *et al.*, "Chainsdi: A software-defined infrastructure for regulation-compliant home-based healthcare services secured by blockchains," *IEEE Systems Journal*, vol. 14, no. 2, pp. 2042–2053, 2019.

[172] J. Xu, K. Xue, S. Li, H. Tian, *et al.*, "Healthchain: A blockchain-based privacy preserving scheme for large-scale health data," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8770–8781, 2019.

[173] D. Ichikawa, M. Kashiyama, and T. Ueno, "Tamper-resistant mobile health using blockchain technology," *JMIR mHealth and uHealth*, vol. 5, no. 7, p. e111, 2017.

[174] A. Zhang and X. Lin, "Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain," *Journal of medical systems*, vol. 42, no. 8, pp. 1–18, 2018.

[175] P. Li, S. D. Nelson, B. A. Malin, and Y. Chen, "Dmms: A decentralized blockchain ledger for the management of medication histories," *Blockchain in healthcare today*, vol. 2, 2019.

[176] M. Shen, Y. Deng, L. Zhu, X. Du, *et al.*, "Privacy-preserving image retrieval for medical iot systems: A blockchain-based approach," *IEEE Network*, vol. 33, no. 5, pp. 27–33, 2019.

[177] A. E. B. Tomaz, J. C. Do Nascimento, A. S. Hafid, and J. N. De Souza, "Preserving privacy in mobile health systems using non-interactive zero-knowledge proof and blockchain," *IEEE Access*, vol. 8, pp. 204441–204458, 2020.

[178] P. Labs, "Interplanetary file system," 2021. https://ipfs.io/.

[179] Storj, "Storj decentralized cloud storage," 2021. https://www.storj.io/.

[180] Swarm, "Swarm: Decentralized storage and communication system," 2021. https://www.ethswarm.org/.

[181] J. L. Roberts, S. Pereira, and A. L. McGuire, "Should you profit from your genome?," *Nature Biotechnology*, vol. 35, no. 1, pp. 18–20, 2017.

[182] Shivom, "A global genomics ecosystem based on distributed ledger technology," 2018. https://www.shivom.io/.

[183] D. Grishin, K. Obbad, P. Estep, K. Quinn, *et al.*, "Accelerating genomic data generation and facilitating genomic data access using decentralization, privacy-preserving technologies and equitable compensation," *Blockchain in Healthcare Today*, vol. 1, pp. 1–23, 2018.

[184] R. Kain, S. Kahn, D. Thompson, D. Lewis, *et al.*, "Database shares that transform research subjects into partners," *Nature biotechnology*, vol. 37, no. 10, pp. 1112–1115, 2019.

[185] LunaPBC, "Lunadna is approved by the sec to offer ownership shares to individuals for sharing data," 2018. https://www.lunadna.com/ownership-shares-for-sharing-data-2/.

[186] DNAsimple Inc., "Dna simple," 2016. https://www.dnasimple.org/newhome.

[187] F. K. Dankar, "Data privacy through participant empowerment," *Nature Computational Science*, vol. 1, no. 3, pp. 175–176, 2021.

[188] J. Zhang, N. Xue, and X. Huang, "A secure system for pervasive social network-based healthcare," *Ieee Access*, vol. 4, pp. 9239–9250, 2016.

[189] S. Padhye, R. A. Sahu, and V. Saraswat, *Introduction to cryptography*. CRC Press, 2018.

[190] E. Ayday, Q. Tang, and A. Yilmaz, "Cryptographic solutions for credibility and liability issues of genomic data," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 1, pp. 33–43, 2017.

[191] T.-T. Kuo, J. Kim, and R. A. Gabriel, "Privacy-preserving model learning on a blockchain network-of-networks," *Journal of the American Medical Informatics Association*, vol. 27, no. 3, pp. 343–354, 2020.

[192] R. Cramer, I. B. Damgård, *et al.*, *Secure multiparty computation*. Cambridge University Press, 2015.

[193] F. McKeen, I. Alexandrovich, I. Anati, D. Caspi, *et al.*, "Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave," in *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016*, pp. 1–9, 2016.

[194] D. Grishin, K. Obbad, P. Estep, M. Cifric, *et al.*, "Blockchain-enabled genomic data sharing and analysis platform—v4. 52," *Nebula Genomics*, 2018.

[195] W. Zheng, Y. Wu, X. Wu, C. Feng, *et al.*, "A survey of intel sgx and its applications," *Frontiers of Computer Science*, vol. 15, no. 3, pp. 1–15, 2021.

[196] HL7, "Fast healthcare interoperability resources (fhir)," 2019. https://www.hl7.org/fhir/.

[197] S. Cao, X. Zhang, and R. Xu, "Toward secure storage in cloud-based ehealth systems: a blockchain-assisted approach," *IEEE Network*, vol. 34, no. 2, pp. 64–70, 2020.

[198] T.-T. Kuo, R. A. Gabriel, and L. Ohno-Machado, "Fair compute loads enabled by blockchain: sharing models by alternating client and server roles," *Journal*

of the American Medical Informatics Association, vol. 26, no. 5, pp. 392–403, 2019.

[199] T.-T. Kuo, R. A. Gabriel, K. R. Cidambi, and L. Ohno-Machado, "Expectation propagation logistic regression on permissioned block chain (explorerchain): decentralized online healthcare/genomics predictive model learning," *Journal of the American Medical Informatics Association*, vol. 27, no. 5, pp. 747–756, 2020.

[200] E. R. Riggs, D. R. Azzariti, A. Niehaus, S. R. Goehringer, *et al.*, "Development of a consent resource for genomic data sharing in the clinical setting," *Genetics in Medicine*, vol. 21, no. 1, pp. 81–88, 2019.

[201] LunaPBC, "Luna dna," 2018. https://www.lunadna.com/lunapbc/.

[202] G. Gürsoy, R. Bjornson, M. E. Green, and M. Gerstein, "Using blockchain to log genome dataset access: efficient storage and query," *BMC medical genomics*, vol. 13, no. 7, pp. 1–9, 2020.

[203] S. Ma, Y. Cao, and L. Xiong, "Efficient logging and querying for blockchain-based cross-site genomic dataset access audit," *BMC Medical Genomics*, vol. 13, no. 7, pp. 1–13, 2020.

[204] GDPR.EU, "Complete guide to gdpr compliance," 2021. https://gdpr-info.eu/art-17-gdpr/.

[205] N. Mamo, G. M. Martin, M. Desira, B. Ellul, *et al.*, "Dwarna: a blockchain solution for dynamic consent in biobanking," *European Journal of Human Genetics*, vol. 28, no. 5, pp. 609–626, 2020.

[206] X. Wang, Y. Zhao, and F. Pourpanah, "Recent advances in deep learning," 2020.

[207] L. Lyu, J. Yu, K. Nandakumar, Y. Li, *et al.*, "Towards fair and privacy-preserving federated deep models," *IEEE Trans. on Parallel and Distributed Systems*, vol. 31, no. 11, pp. 2524–2541, 2020.

[208] A. Shafee and T. A. Awaad, "Privacy attacks against deep learning models and their countermeasures," *Jour. of Sys. Arch.*, p. 101940, 2020.

[209] E. S. Lander, L. M. Linton, B. Birren, C. Nusbaum, *et al.*, "Initial sequencing and analysis of the human genome," 2001.

[210] D. Almojil, Y. Bourgeois, M. Falis, I. Hariyani, *et al.*, "The structural, functional and evolutionary impact of transposable elements in eukaryotes," *Genes*, vol. 12, no. 6, p. 918, 2021.

[211] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. of the 22nd ACM SIGSAC conf. on comp. and commun. security*, pp. 1310–1321, 2015.

[212] Y. Aono, T. Hayashi, L. Wang, S. Moriai, *et al.*, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. on Info. Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[213] P. Li, J. Li, Z. Huang, T. Li, *et al.*, "Multi-key privacy-preserving deep learning in cloud computing," *Future Gener. Comp. Sys.*, vol. 74, pp. 76–85, 2017.

[214] T. T. Phuong *et al.*, "Privacy-preserving deep learning via weight transmission," *IEEE Trans. on Info. Forensics and Security*, vol. 14, no. 11, pp. 3003–3015, 2019.

[215] Y. Li, C. Chen, N. Liu, H. Huang, *et al.*, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2020.

[216] Y. Lu, X. Huang, Y. Dai, S. Maharjan, *et al.*, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Trans. on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2019.

[217] Y. Qi, M. S. Hossain, J. Nie, and X. Li, "Privacy-preserving blockchain-based federated learning for traffic flow prediction," *Future Gener. Comp. Sys.*, vol. 117, pp. 328–337, 2021.

[218] G. Xu, H. Li, S. Liu, K. Yang, *et al.*, "Verifynet: Secure and verifiable federated learning," *IEEE Trans. on Info. Forensics and Security*, vol. 15, pp. 911–926, 2019.

[219] F. Ouyang, *Digital Communication for Practicing Engineers.* John Wiley & Sons, 2019.

[220] S. Akhter, B. A. Bailey, P. Salamon, R. K. Aziz, *et al.*, "Applying shannon's information theory to bacterial and phage genomes and metagenomes," *Scientific reports*, vol. 3, no. 1, pp. 1–7, 2013.

[221] J. T. Machado, "Information analysis of the human dna," *Nonlinear Dynamics*, vol. 98, no. 4, pp. 3169–3186, 2019.

[222] D. Fiore, R. Gennaro, and V. Pastro, "Efficiently verifiable computation on encrypted data," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 844–855, 2014.

[223] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "Cloud-assisted multiparty computation from fully homomorphic encryption.," *IACR Cryptol. ePrint Arch.*, vol. 2011, p. 663, 2011.

[224] G. Asharov, A. Jain, A. López-Alt, E. Tromer, *et al.*, "Multiparty computation with low communication, computation and interaction via threshold fhe," in *Annual Inter. Conf. on the Theory and Appli. of Crypt. Techni.*, pp. 483–501, Springer, 2012.

[225] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM Trans. on Computation Theory*, vol. 6, no. 3, pp. 1–36, 2014.

[226] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-lwe and security for key dependent messages," in *Annual cryptology conference*, pp. 505–524, Springer, 2011.

[227] J. Chen and S. Micali, "Algorand: the efficient and democratic ledger," *arXiv preprint arXiv:1607.01341*, 2016.

[228] J. Zou, M. Huss, A. Abid, P. Mohammadi, *et al.*, "A primer on deep learning in genomics," *Nature genetics*, vol. 51, no. 1, pp. 12–18, 2019.

[229] PALISADE, "Palisade lattice cryptography library," 2021. https://palisade-crypto.org/.

[230] A. De Caro and V. Iovino, "jpbc: Java pairing based cryptography," in *2011 IEEE sympo. on comp. and commun.*, pp. 850–855, IEEE, 2011.

[231] ENCODE, "Encode: Encyclopedia of dna elements," 2021. https://www.encodeproject.org/.

[232] Y. Zhang, S. Qiao, S. Ji, and Y. Li, "Deepsite: bidirectional lstm and cnn models for predicting dna–protein binding," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 4, pp. 841–851, 2020.

[233] B. McMahan, E. Moore, D. Ramage, S. Hampson, *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelli. and stat.*, pp. 1273–1282, PMLR, 2017.