On Secure Cloud Computing for Genomic Data: From Storage to Analysis

THÈSE N^{O} 8367 (2018)

PRÉSENTÉE LE 13 AVRIL 2018

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE POUR LES COMMUNICATIONS INFORMATIQUES ET LEURS APPLICATIONS 1
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Zhicong HUANG

acceptée sur proposition du jury:

Prof. A. Lenstra, président du jury
Prof. J.-P. Hubaux, Prof. J. Fellay, directeurs de thèse
Dr D. Baker, rapporteuse
Dr P. Flicek, rapporteur
Prof. B. Ford, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2018

Dark clouds become heaven's flowers when kissed by light.
Stray Birds, by Rabindranath Tagore, 1916

To my family and Danhua

Abstract

Although privacy is generally considered to be the right of an individual or group to control information about themselves, such a right has become challenging to protect in the digital era, this is exemplified by the case of cloud-based genomic computing. Despite the rapid progress in understanding, producing, and using genomic information, the practice of genomic data protection remains a fairly underdeveloped area. One of the indisputable reasons is that most nonexpert individuals do not realize the sensitive nature of their genomic data, unless it has been used against them. Many commercial organizations take advantage of their customers by taking control of personal genomic information, if customers want to benefit from services such as genetic analysis; even worse, these organizations often do not enforce proper protection, which could result in embarrassing data breaches. In this thesis, we investigate the potential threats of cloud-based genomic computing systems and propose various countermeasures by taking into account the functionality requirement.

We begin with the most basic system where only symmetric encryption is needed for the cloud storage of genomic data, and we propose a new solution that protects the data against brute-force attacks that threaten the security of password-based encryption in direct-to-consumer companies. The solution employs honey encryption, where plaintext messages need to be transformed to a different space with uniform distribution on elements. We present a novel distribution-transformation encoder. We provide formal security proof of our solution.

We analyze the scenario where efficient searching on encrypted data is necessary. We propose a system that provides fast retrieval on encrypted compressed data and that enables individuals to authorize access to fine-grained regions during data retrieval. Our solution addresses three critical dimensions in platforms that use large genomic data: encryption, compression, and efficient data retrieval. Compared with a previous de facto standard solution for storing aligned genomic data, our solution uses 18% less storage.

To enable complicated data analysis, we focus on a proposal for secure quality-control of genomic data by using secure multi-party computation based on garbled circuits. Our proposal is for aggregated genomic data sharing, where researchers want to collaborate to perform large-scale genome-wide association studies in order to identify significant genetic variants for certain diseases. Data quality control is the very first stage of such a collaboration and remains a driving factor for further steps. We investigate the feasibility of advanced cryptographic techniques in the data protection of this phase. We demonstrate that for certain protocols, our solution is efficient and scalable.

With the advent of precision medicine based on genomic data, the future of big data

has become clearly inseparable from cloud-based genomic computing. It is important to continuously re-evaluate the standards of cloud-based genomic computing as novel technologies are developed, security threats arise, and more complex genomic analyses become possible. This is not only a battle against cyber criminals, but also against rigid and ignorant practices. Integrative solutions that carefully consider the use and misuse of personal genomic data are essential for ensuring secure, effective storage and maximizing utility in treating and preventing disease.

Keywords : genomic privacy, cloud computing, brute-force attacks, password-based encryption, honey encryption, compression, order-preserving encryption, secure multi-party computation, differential privacy, garbled circuits

Résumé

Être en mesure de protéger sa vie ou sphère privée en contrôlant la divulgation de ses données personnelles est traditionnellement considéré comme un droit. Malheureusement, cet acquis est devenu difficile à protéger dans l'ère numérique. Il en va particulièrement ainsi pour les données génomiques, sur lesquelles des calculs doivent souvent être exécutés directement dans le "cloud". En dépit des rapides progrès scientifiques dans la compréhension, la production et l'usage des données génomiques, la protection de celles-ci reste un sujet relativement peu développé. La méconnaissance des enjeux dépendants de la protection de ces données, dont font preuve la plupart des non-initiés, en est l'une des raisons. Plusieurs entreprises proposent, par exemple, des analyses génétiques à leurs clients et profitent de leur relative ignorance pour utiliser leurs données génomiques. De façon plus inquiétante encore, certaines de ces organisations ne protègent pas correctement les données récoltées, les rendant vulnérables à des vols. Dans cette thèse, nous étudions les potentielles menaces sur les systèmes de stockage et traitement de données génomiques basés sur une approche "cloud-computing", et proposons plusieurs solutions afin de les protéger tout en respectant leurs impératifs en terme de fonctionnalités et performances.

Sur la base d'un système basique dans lequel les données sont simplement protégées à l'aide d'un chiffrement symétrique avant d'être stockées dans le "cloud", nous proposons une nouvelle solution prévenant contre les attaques de type "brute-force" qui menacent habituellement la sécurité du chiffrement par mot de passe. Cette solution repose sur le chiffrement "honey", dans lequel les messages en clair sont préalablement transformés de manière à suivre une distribution uniforme. Nous pr
'esentons une nouvelle transformation de distribution et donnons une preuve formelle de la sécurité de notre solution. Nous nous intéressons ensuite à un scénario dans lequel une méthode de recherche efficace dans des données chiffrées est nécessaire. Nous proposons un système capable de retrouver rapidement un élément dans des données chiffrées et compressées, tout en permettant aux utilisateurs d'en limiter l'accès de manière granulaire. Notre solution est, au meilleur de nos connaissances, la première à regrouper les trois éléments clés des plateformes de stockage de données génomiques: chiffrement, compression et efficacité de recherche. Nous montrons notamment que notre solution réduit l'espace de stockage requis de 18% par rapport aux standards existants.

Afin de permettre des analyses complexes de données, nous proposons une solution utilisant les techniques dites "secure multi-party computation" basées sur les "garbled circuits" afin de permettre un contrôle de la qualité des données génomiques distribuées sur de multiples sites, tout en respectant leur nature sensible. De tels contrôles sont

une étape cruciale, préalable à toute étude d'association des génomes (GWAS). Notre solution permet donc la collaboration entre chercheurs sur ces études en rendant possible le partage de données génomiques agrégées, afin d'isoler des variations génétiques intéressantes dans l'étude de certaines maladies. Nous testons la faisabilité de ces solutions lorsque des techniques cryptographiques récentes sont employées afin de protéger les données et montrons quels parties de cette phase de contrôle peuvent en bénéficier.

Avec les avancées de la médecine de précision basée sur les données génomiques, le futur du "big data" apparaît inexorablement lié au traitement massif des données génomiques dans le "cloud". Il est donc important de continuellement réévaluer les standards et pratiques dans ce secteur. En effet, la technologie se développe et des analyses de plus en plus complexes deviennent possibles, mais ce progrès est aussi accompagné de nouvelles menaces contre la sécurité de ces données sensibles. Il s'agit d'un combat, non seulement contre les pirates informatiques, mais aussi contre l'ignorance et les mauvaises pratiques en matière de sécurité. Il est essentiel que des solutions intégrables et considérant la possibilité d'un usage abusif des données soient proposées, afin d'assurer la sécurité, l'efficacité du stockage, et la maximisation de l'utilité des données dans la prévention et le traitement des maladies.

Mots-clés : protection des données génomiques, calculs dans le "cloud", attaque de "brute-force", chiffrement "honey", compression, chiffrement respectant l'ordre, "secure multi-party computation", "differential privacy", "garbled circuits"

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisor, Prof. Jean-Pierre Hubaux, for the continuous support of my Ph.D. study and related research, for his patience, guidance, and immense knowledge. After 5 years, I still have the fresh memory of my first day stepping into this lab, with limited research experience and lack of initiative. His guidance and encouragement helped me in all the time of research and writing of this thesis. Beyond that, I am extremely grateful for a lifelong lesson of which he keeps reminding me: Be proactive and express myself!

I am particularly grateful to my co-advisor, Prof. Jacques Fellay, for his instruction, insights and useful critiques in all my research problems. Being devoted to a cross-discipline research topic, I could not have completed this work without his support and knowledge from a domain that I was not familiar with at the beginning.

My sincere thanks also goes to the rest of my thesis committee: Dr. Dixie Baker, Dr. Paul Flicek, Prof. Bryan Ford, and Prof. Arjen Lenstra, who invested valuable time and effort in reviewing this thesis, and some of whom travelled a long way for the defense of my thesis.

I would like to thank my co-authors for our fruitful collaborations and their valuable contributions to this thesis: Prof. Erman Ayday, Dr. Raeka S. Aiyar, Prof. Ari Juels, Prof. Zoltán Kutalik, Dr. Huang Lin, Dr. Adam Molyneaux, Prof. Lars M. Steinmetz, and Dr. Zhenyu Xu. In addition, I have the privilege to work with many other brilliant minds in other joint works that are not presented in this thesis, and great thanks to them: Dr. Jean Louis Raisaro, Florian Tramèr, David Froelicher, Patricia Egger, Joao S. Sousa, Christian Mouchet, Cédric Lefebvre, Prof. Carlos Aguilar-Melchor, Prof. Marc-Olivier Killijian, Dr. Sahel Shariati Samani, Prof. Mark Elliot, Dr. Mathias Humbert, Dr. Alexandra Olteanu, and Prof. Karl Aberer. I would like to also thank Dr. Kristin Lauter for offering me the great internship in Microsoft Research, where I was fortunate to collaborate with many excellent researchers: Dr. Kim Laine, Dr. Hao Chen, Dr. Ran Gilad-Bachrach, Kyoohyung Han, and Amir Jalali. My special thanks are extended to the secretaries and system administrators for all their great support and meticulous work during the 5 years of my Ph.D. study, and to Holly Cogliati-Bauereis who helps me grow from a rookie writer to a somewhat-proficient writer in English.

I wish to thank all my colleagues and friends in EPFL for the joys and tears we shared together in this grand journey: Jean Louis, Anh, Italo, Juan, Alexandra, Mathias, Joao, David, Christian, Ludovic, Qiang, Yan, Tong, Junrui, Fengyun, Bin, Kaicheng, Li, Tao, Wouter, Bogdan, and Kévin. A very special gratitude goes out to Jean Louis who has been a great friend in these 5 years, and who has brought me all the memorable

experiences in social life! As another special mention, my Ph.D. life abroad would not have been as pleasant without the friendship of more than 10 years from these people outside EPFL: Jinle, Zhiqian, Guiyu, Ke, Jie, Wenlong, Yadan, Xiaomin, Tingting, Lixing, and Yinmin.

I dedicate this thesis to my parents for their unconditional support and love, and for their care for me while I study abroad on the other side of the continent. Last but not least, my heartfelt thanks goes to Danhua, for her love and support during the past 8 years, for her understanding in our 5-year cross-country relationship, and for the happiness she brings into my life. To Danhua: There is no one else I would like to spend the rest of my life with, so, will you marry me?

Contents

Chapter 1

Introduction

Due to major advances in genomic research and to the plummeting cost of high-throughput sequencing, the use of human genomic data is rapidly expanding in various domains, including healthcare (e.g., genomic-based personalized medicine), research (e.g., genome-wide association studies), direct-to-consumer (DTC) services (e.g., ancestry determination), legal cases (e.g., paternity tests), and forensics (e.g., criminal investigation). It is now possible for physicians to adjust the prescription of certain drugs based on the genetic makeup of their patients, for individuals to learn about their genetic predisposition to serious diseases, and for couples to find out if their potential offspring has an increased likelihood of developing rare genetic diseases. These exciting use cases come with tremendous economic potentials, which explains the success of many enterprises such as *23andMe* and *Color Genomics*, and new services from major stakeholders such as Google Genomics [1], IBM Watson [2], Microsoft Genomics [3], Apple Research Kit [4], and Amazon AWS Genomics [5].

However, such a vast exploitation of genomic data raises critical privacy issues. Because genomic data includes valuable and sensitive information about individuals, leakage of such data can have serious consequences, including discrimination (e.g., by a potential employer), denial of services due to genetic predisposition (e.g., by an insurance company), or even blackmail (e.g., using sensitive paternity information). With the technology becoming increasingly mature, these privacy concerns could lead to the humanity nightmare that was depicted by the 1997 science fiction movie *Gattaca*. To benefit from the aforementioned interesting biomedical services, individuals and even large medical institutions tend to hand over the control of their data to third-party stakeholders who might or might not protect the data properly. Although it is generally true that large cloud-computing companies have better capabilities in protecting their system and data, data breaches happen from time to time, either because the data is not properly encrypted or the key is stolen. For example, in 2013, 1 billion user accounts were hacked in Yahoo's database, where most of the information is not encrypted, including even security questions that could be used to reset passwords [6]. Ironically, because large databases are valuable attack targets, the world's biggest data breaches often happen in well-known companies that should have protected the data better than others [7]. If the same failure happened with a genomic database, the consequences would be much worse. As the security expert Bruce Schneier once said [8],

> *If someone steals your password, you can change it. But if someone steals your thumbprint, you can't get a new thumb. The failure modes are very different.*

The argument holds for personal genomic data because this data is unique to every human being. Even worse, users cannot control how companies will deal with their personal genomic data after they sign general consent forms, e.g., when *23andMe* sold customers' genetic data to another biotechnology company *Genentech* for 60 million dollars [9].

Unfortunately, existing practices could lead to the abuse of genomic data; this abuse is driven purely by economic interests. Indeed, due to the cost of security and privacy, a privacy-preserving implementation might impair the performance and utility, which is the reason why enterprises and institutions tend to go with easy and efficient solutions for processing data. Historically, the initially insecure Internet has indeed survived from numerous attacks, thanks to the gradually improving security deployment (e.g., from *HTTP* to *HTTPS*). Nevertheless, in the case of genomic data, as mentioned above, the failure modes are different; once an individual's genomic information is exposed, the corresponding privacy is lost forever. Therefore, it is crucial to store and manage genomic data in a privacy-preserving way in order to enable its secure use from the beginning.

Existing mechanisms for protecting the privacy of genomic data include (i) anonymization, which has proven to be ineffective for genomic data [10, 11, 12], (ii) adding noise to published genomic data or statistics for medical research (e.g., to guarantee differential privacy [13, 14, 15, 16]), (iii) cryptography (e.g., homomorphic encryption [17, 18], private set intersection [19], etc.), and (iv) trusted hardware ([20, 21, 22]).

In this thesis, in order to enable the secure use of genomic data, we propose various defense mechanisms that depend on the functionality requirement, . These privacy-conscious systems can de deployed under different scenarios, ranging from storage-only applications to complicated statistical analysis. We implement these systems and demonstrate that, with reasonable performance costs, they are scalable in real-life applications with strong security and privacy guarantee.

1.1 Related Work

Because genomic information is highly personal, privacy has become a major concern as these data become more widely generated, disseminated, and unwillingly exposed [113, 114]. For example, coarse-grained encryption and access control to genomic data could lead to incidental findings that doctors often prefer to avoid [115]. Standard sample de-identification has been proven insufficient for complete protection of genetic privacy [25]. Establishing a secure and privacy-preserving solution for genomic data storage is urgently needed to facilitate the usage and transfer of the data. For example, storing sequence data on a cloud is an attractive option, considering the size and the required availability of the data [116, 117, 118]. However, access threats in this case are even more serious because the data owner has to trust insiders on the cloud (e.g., the cloud administrator or high-privileged system software). These concerns around genomic data and the corresponding countermeasures have been extensively investigated by researchers in recent years [23].

1.1.1 Statistical inference attacks and differential privacy

Homer *et al.* [11] show the possibility of inferring the participation of an individual in a genotype database with the help of public allele frequencies. This work, known as *Homer's attack*, became a landmark for genomic privacy, due to its resulting widespread fear that even sharing aggregate statistics is no longer safe. Afterwards, the NIH even removed aggregate genomic data from open-access databases and urged the scientific community to take precautions before sharing any aggregate GWAS data. Wang *et al.* [12] give similar results of inference power based on p-values released in genome-wide association studies. Various other statistical inference attacks have been proposed, which makes the situation of genomic privacy more worrisome. These two attacks are typical examples for the general category of statistical inference attacks on aggregate genomic data, and a plethora of similar attacks have been proposed in the literature [134, 136, 137, 135, 133]. Hence, we wonder if it is even possible to protect against these attacks. In response to the above concerns about published genomic statistics, Fienberg *et al.* [13] propose to apply Laplacian noise to the released data to achieve differential privacy. Another approach, using an exponential mechanism, to achieving differential privacy in a genome-wide association study, is proposed by Johnson and Shmatikov [14]. Yu *et al.* [15] present scalable privacy-preserving methods in genome-wide association studies that are based on the Laplacian mechanism and an exponential mechanism. To further improve the accuracy of the shared noisy data, Tramèr et al. [16] propose to relax differential privacy by bounding the prior knowledge of potential attackers. In spite of these works on differentially private genomic data, Fredrikson *et al.* [24] demonstrate an unsatisfactory tradeoff between privacy and utility in an end-to-end case study of personalized dosing of warfarin: after enforcing recommended level of differential privacy, the result is too noisy to be used. A similar unsatisfactory result in an association study is also mentioned by Erlich and Narayanan [25].

1.1.2 Computation on encrypted sensitive data

Apart from the family of solutions based on differential privacy, which turns out to be criticized heavily by practitioners, many promising systems have been proposed in the cryptography community. Jha *et al.* [26] design several privacy-preserving protocols for some fundamental genomic computations (edit distance and Smith-Waterman score) that use oblivious transfer and oblivious circuit evaluation. Kantarcioglu *et al.* [18] propose the use of homomorphic encryption to store encrypted genomic sequence records in a centralized repository and in a way that queries can be executed without decryption and thus without violating participants' privacy. Baldi *et al.* [19] propose a set of techniques based on private set operations to address genomic privacy in several important applications, namely, paternity tests, personalized medicine, and genetic compatibility tests. To store and retrieve raw genomic data in a privacy-preserving manner, Ayday *et al.* [17] introduce a framework that integrates stream ciphers and order-preserving encryption. Other researchers also propose to protect privacy in genomic computation by partitioning the computation through program specialization, according to the sensitivity levels of different parts of the genome data [27]. The improvement of advanced cryptographic techniques has helped to advance many other secure cloud computing solutions [28, 29, 30, 31, 32, 33, 34, 35, 36] that share similar cloud models to some extent with the aforementioned systems. Nevertheless, in most cases, these solutions are based

on heavy public-key cryptography, such as homomorphic encryption and garbled circuits, which demands further research before they can be widely deployed in practice.

1.1.3 Trusted computing

In another direction, some researchers have begun to explore the possibility of employing trusted hardware for secure genomic computing in public clouds, thanks to several breakthroughs in trusted computing, such as Intel's *Software Guard eXtension* (SGX). The most evident benefit of solutions in this dimension is the significant performance gain, compared to the above advanced cryptographic solutions. Under the same system and threat models, solutions have been proposed to address different genomic computing problems [20, 21, 22]. In addition, several distributed computing systems based on Intel SGX have been designed to solve more general big-data-processing problems [37, 38, 39]. It would be indeed encouraging if the same security and privacy could be achieved with trusted hardware instead of heavy public-key cryptography; however, many researchers raise concerns about its trust model and side-channel information leakage. For example, multiple research groups show that it is practical to extract sensitive data from CPU cache even though the main memory is encrypted [40, 41, 42, 43]. Moreover, attacking the Intel processors was escalated in the start of 2018 [44], when several groups independently found serious CPU vulnerabilities in Intel's hardware [45, 46], further raising concerns about the security promise of SGX. Therefore, it is unlikely that trusted computing will be able to supplant advanced cryptographic solutions under all circumstances.

1.1.4 Honey encryption

There have been a number of practices of applying deception and decoys in the literature of computer security. Honeypots [86] are fake computer systems intended to bait malicious actions that will be tracked and studied once these systems are probed or compromised. Honeypots are widely used in intrusion detection system [87, 88, 89]. Similarly, a honeynet [90] is proposed to assist the system administrator in identifying malicious traffic on the enterprise network.

The Kamouflage system [91] and honeywords [92] are designed to protect a password vault by constructing plausible decoy passwords. Juels and Ristenpart [61] formalize such a construction process with the concept of distribution-transforming encoder (DTE), and propose honey encryption that provides security beyond the brute-force bound of password-based encryption. As a formal extension to the above construction, Chatterjee et al. [93] propose a design, called NoCrack, for crack-resistant password vaults using honey encryption with natural language encoders. NoCrack shares the same motivation with Chapter 2 of this thesis: When an attacker decrypts with the wrong password, it gets a plausible-looking decoy, such that it does not know whether the password is indeed wrong or not.

We improve honey encryption in this thesis by proposing a novel and useful DTE, and apply the result to the protection of genomic data. To the best of our knowledge, prior to this thesis, no cryptographic solution in genomic privacy addresses the challenge of long-term threats to encryption, such as quantum computing [82], or of the common short-term threat of brute-force cracking of password-based encryption [83, 84, 85].

1.2 Contributions

In this thesis, we investigate the problem of **secure cloud computing for genomic data**. First, we start from the most basic system where only symmetric encryption is needed for the storage of genomic data, then we propose a new solution that protects the data against brute-force attacks that threaten the security of password-based encryption in direct-to-consumer companies. Second, we analyze the scenario where efficient searching on encrypted data is necessary. We propose a system that provides fast retrieval on encrypted compressed data and that enables individuals to authorize access to fine-grained regions during data retrieval. Finally, we explore various methods for enabling readers to design privacy-conscious systems for complicated data analysis, with the focus on a proposal for secure quality control of genomic data by using secure multi-party computation based on garbled circuits.

Our contributions are as follows:

1. We propose a novel technique for securing genomic data against data breaches that involve a computationally unbounded adversary (an essential requirement given the longevity of genomic data). This is particularly useful in the case of password-based encryption that is frequently used by direct-to-consumer services, where encryption keys are derived from low-entropy user-chosen passwords. The solution employs a symmetric encryption primitive called *honey encryption*, where plaintext messages need to be transformed to a different space with uniform distribution on elements. Such a transformation needs to be provably secure and efficient. We design and analyze several distribution and transformation models for genome sequences, and we present a formal security analysis of our proposed techniques. In addition, we propose and analyze techniques for preventing an adversary from exploiting side information (physical traits of victims) in order to decrypt genomes. We implement our solution in Python and show that it is efficient, especially under a parallel-processing environment.

2. In clinical genomics, the continuous evolution of bioinformatic algorithms and sequencing platforms makes it possible to store patients' complete aligned genomic data in addition to variant calls relative to a reference sequence. Due to the large size of human genome-sequence data files (varying from 30 GB to 200 GB depending on coverage), genomics laboratories face two major challenges: the costs of storage and the efficiency of the initial data processing. We present a privacy-preserving solution for the secure storage of compressed aligned genomic data. Our solution enables selective retrieval of encrypted data and improves the efficiency of downstream analysis (e.g., variant calling). Compared with a previous *de facto* standard solution for storing aligned genomic data, our solution uses 18% less storage. Our solution maintains both efficient compression and downstream data processing, and it provides unprecedented levels of security in genomic data storage. Compared with previous work, the distinguishing features of our solution are that (1) it is position-based instead of read-based, and (2) it enables random querying of a sub-region in an encrypted file. Our method thus offers a space-saving, privacy-preserving, and effective solution for the storage of clinical genomic data.

3. Due to the limited power of small-scale genome-wide association studies (GWAS), in order to perform large-scale GWAS, researchers tend to collaborate and establish

a larger consortium. Genome-wide association meta-analysis (GWAMA) is a statistical tool that synthesizes results from multiple independent studies to increase the statistical power and reduce false-positive findings of GWAS. We propose a secure quality control protocol that enables the quality of data to be checked in a privacy-preserving way without revealing sensitive information to a potential adversary. Our solution employs state-of-the-art cryptographic and statistical techniques for privacy protection. To demonstrate the efficiency and scalability on commodity machines, we implement the solution in a meta-analysis pipeline with real data. The distributed execution of our solution on a cluster of 128 cores for one million genetic variants takes less than one hour, which is a modest cost considering the 10-month time span usually observed for the completion of the QC procedure that includes timing of logistics. To end with a promising future for secure processing of genomic data, we discuss multiple alternative solutions that might provide benefits (e.g., high efficiency, low communication) under slightly different security and trust models.

Before we go into the technical details, it is crucial to understand why the above proposed systems in this thesis are superior to off-the-shelf techniques (e.g., from cryptography) that are directly applicable for general data protection. First of all, some special characteristics of genomic data might require researchers and developers to design new security primitives. For example, the aforementioned failure mode of genomic data poses a long-term protection requirement in cloud storage, which motivates us to design a new encryption scheme in Chapter 2 instead of using a conventional one that is vulnerable to brute-force attacks. Secondly, during the design of these systems, it is imperative to take existing practices into account so that the systems are usable and appreciated by practitioners in the genomic and medical community. This actually prevents us from directly applying off-the-shelf techniques for general data, because genomic data are generally stored in well-structured format and are used in predefined ways that might lead to unintended information leakage (Chapter 3) if the protection framework is not crafted accordingly. Finally, many applications in medical research (e.g., statistical analysis on protected data) demand research on advanced protection techniques, such as secure multi-party computation and homomorphic encryption. During the design of such systems, it is also crucial to combine various techniques for improving performance and privacy, and to prove the security and privacy guarantee of such combination, e.g., the combination of parallel multi-party computation and differential privacy in Chapter 4.

1.3 Thesis Outline

Following the secure cloud-computing scenarios from storage to data analysis, we organize this thesis as follows. In Chapter 2, we discuss the secure cloud storage for genomic data, and propose a detailed encryption that provides resistance against brute-force attacks. In Chapter 3, we extend the scenario to take data searching into account. We describe a solution with efficient selective data retrieval. In Chapter 4, we focus on solutions that encompass complicated data-analysis operations, at a cost of performance overhead, accuracy loss, or weak trust model. We describe a secure multi-party computation solution in detail.

1.4 Publications

Chapter 2 is an extended version of [47]. Chapter 3 is an extension of [48]. Finally, Chapter 4 rests on the results of [49]. I am thankful to the following collaborators who made valuable contribution to these projects: Erman Ayday (Bilkent University), Raeka S. Aiyar (Stanford University), Jacques Fellay (EPFL), Jean-Pierre Hubaux (EPFL), Ari Juels (Cornell Tech), Zoltán Kutalik (University Hospital Lausanne), Huang Lin (Applied Science and Technology Research Institute), Adam Molyneaux (Sophia Genetics), Lars M. Steinmetz (Stanford University), and Zhenyu Xu (Sophia Genetics).

Chapter 2

Securing storage of genomic data

Cloud storage is becoming more and more popular in people's daily life, mostly thanks to its promise of availability in any time and on any device. Moreover, considering the scalability, efficiency and reliability provided by cloud services, many organizations adopt cloud solutions for their IT work instead of building and maintaining the infrastructure by themselves. According to the 2016 HIMSS Analytics Cloud Study [50] in United States, 84% of the surveyed healthcare organizations are currently using cloud services, 25.6% of which fall into the *public* cloud computing model. In spite of privacy concerns, healthcare organizations are ready to adopt public cloud storage, provided that the cloud providers sign Business Associate Agreement (BAA) and offer HIPAA[1]-compliant services, such as Google Cloud Drive, Microsoft OneDrive and Amazon [51]. Nevertheless, when organizations and (especially) individual users place their data with cloud computing services, they lose the complete control of that information. There have been numerous cases and debates about whether cloud providers scan clients' data [52]. For example, Microsoft's SkyDrive (now OneDrive) and Apple's iCloud reserve the right to scan users' private files for content they deem "inappropriate", which led to a number of worrisome results [53, 54]. In this regard, many privacy-centric services start to bring the control back to users while still enabling them to benefit from cloud storage. For instance, *SpiderOak* [55] offers both cloud storage and a security option for users to encrypt data before uploading. In addition, several third-party tools, such as *Boxcryptor* and *Odrive* [56], enhance users with encryption functionality integrated in current popular cloud storage providers (e.g., Google Drive, Dropbox). Finally, even for local data protection, some companies offer software solutions to encrypt the database, such as IBM's DB2 [57] and Microsoft's Access database [58]. In brief, these solutions are mostly based on password-based encryption (PBE), where users choose passwords in order to derive cryptographic keys for standard symmetric encryption such as 128-bit AES. However, since keys are derived from passwords with lower entropy, the security level is actually lower than the 128 bits indicated by the name.

Appropriately designed cryptographic schemes can preserve the utility of data, but they provide security based on assumptions about the computational limitations of ad-

[1]HIPAA (Health Insurance Portability and Accountability Act of 1996) is United States legislation that provides data privacy and security provisions for safeguarding medical information.

versaries. Hence they are vulnerable to brute-force attacks when these assumptions are incorrect or erode over time. Given the longevity of genomic data, serious consequences can result. A genome is (almost) stable over time and thus needs protection over the lifetime of an individual and even beyond, as genomic data is correlated between the members of a single family. It has been shown that the genome of an individual can be probabilistically inferred from the genomes of his family members [59].

In many situations, though, particularly those involving direct use of data by consumers, keys are weak and vulnerable to brute-force cracking *even today*. This problem arises in systems that employ password-based encryption, a common approach to protection of user-owned data. Users' tendency to choose weak passwords is widespread and well documented [60].

Several years ago, Juels and Ristenpart introduced a new theoretical framework for encryption called *honey encryption* (HE) [61]. Honey encryption has the property that when a ciphertext is decrypted with an incorrect key (as guessed by an adversary), the result is a plausible-looking yet incorrect plaintext. Therefore, HE gives encrypted data an additional layer of protection by serving up fake data in response to every incorrect guess of a cryptographic key or password. Notably, HE provides a hedge against brute-force decryption in the long term, giving it a special value in the genomic setting.

However, HE relies on a highly accurate distribution-transforming encoder (DTE) (Section 2.1.2) over the message space. Unfortunately, this requirement jeopardizes the practicality of HE. To enable its use in any scenario, we have to understand the corresponding message space quantitatively, that is, the precise probability of every possible message. When messages are not uniformly distributed, characterizing and quantifying the distribution is a highly non-trivial task. Building an efficient and precise DTE is the main challenge when extending HE to a real use case, and it is what we do in this chapter. Hopefully, the techniques proposed are not limited to genomic data; they are intended to inspire those who want to apply HE to other scenarios, typically when the data shares similar characteristics with genomic data.

In this chapter, we propose to address the problem of protecting genomic data by combining the idea of honey encryption with the special characteristics of genomic data in order to develop a secure genomic data storage (and retrieval) technique that is (i) robust against potential data breaches, (ii) robust against a computationally unbounded adversary, and (iii) efficient.

In the original paper [61], Juels and Ristenpart propose specific HE constructions that rely on existing generation algorithms (e.g. for RSA private keys), or operate over very simple message distributions (e.g., credit card numbers). These constructions, however, are inapplicable to plaintexts with considerably more complicated structure, such as genomic data. Thus substantially new techniques are needed in order to apply HE to this case. Additional complications arise when the correlation between the genetic variants (on the genome) and phenotypic side information are taken into account. This chapter is devoted mainly to addressing these challenges.

2.0.1 Overview

We propose a scheme called GenoGuard. In GenoGuard, genomic data is encoded, encrypted under a patient's password[2], and stored at a centralized biobank. We propose a novel tree-based technique to efficiently encode (and decode) the genomic sequence to meet the special requirements of honey encryption. Legitimate users of the system can retrieve the stored genomic data by typing their passwords.

A computationally unbounded adversary can break into the biobank protected by GenoGuard, or remotely try to retrieve the genome of a victim. The adversary could exhaustively try all the potential passwords in the password space for any genome in the biobank. However, for each password he tries, the adversary will obtain a plausible-looking genome without knowing whether it is the correct one. We also consider the case when the adversary has side information about a victim (or victims) in terms of his physical traits. In this case, the adversary could use genotype-phenotype associations to determine the real genome of the victim. GenoGuard is designed to prevent such attacks, hence it provides protections beyond the normal guarantees of HE.

GenoGuard is highly efficient and can be used by the service providers that offer DTC services (e.g., 23andMe) to securely store the genomes of their customers. It can also be used by medical units (e.g., hospitals) to securely store the genomes of patients and to retrieve them later for clinical use.

2.1 Background

In this section, we briefly introduce some basic concepts of genomics, as well as the honey encryption scheme [61]. To facilitate future references, frequently used notation is listed in Table 2.1.

2.1.1 Genomics

Genetic Locus, Allele, and Single Nucleotide Variant

In this chapter, we consider a genetic **locus** (plural **loci**) as a position on a chromosome. One of a number of alternative forms at a given locus is called an **allele**. Most of the genome is conserved, in comparison to the reference human sequence, in any given individual. The most abundant type of genetic variants are **single nucleotide variants** (SNVs), in which different alleles are observed at the same chromosomal position. Only about 4 million SNVs are observed per individual; they represent the sensitive information that should be protected. In most cases, there are two alleles at a locus, a **major allele**, which is observed with a high frequency in the population, and a **minor allele**, which is observed with low frequency. The frequency of an allele in a given population is denoted as the **allele frequency** (AF). An allele takes a value from the set $\{A, T, C, G\}$. We represent a major allele as 0, and a minor allele as 1. Human chromosomes are inherited in pairs, one from the father and the other from the mother, hence each SNV position has a pair of alleles (nucleotides). For example, the i-th SNV (on the DNA sequence) can be represented as $\text{SNV}_i = xy$, where x (and y) is an allele. As the ordering of x and y does not matter, we represent the value of an SNV_i from the set $\{0, 1, 2\}$,

[2]A patient can choose a low-entropy password that is easier for him/her to remember, which is a common case in the real world [60].

$\mathcal{M}$	sequence (plaintext) space
M	a sequence (message), $\mathsf{M} \in \mathcal{M}$
n	number of SNVs in M
$\mathcal{S}$	seed space
$\mathcal{K}$	key space
$\mathcal{C}$	ciphertext space
p_k	key (password) distribution
p_m	original message distribution
p_d	DTE message distribution
h	storage overhead parameter
$\mathcal{A}$	the adversary against the DTE scheme
$\mathrm{Adv}^{\mathrm{dte}}_{\mathsf{DTE},p_m}(\mathcal{A})$	adversary $\mathcal{A}$'s advantage of distinguishing p_m from p_d
$\mathcal{B}$	the adversary against the HE scheme
$\mathrm{Adv}^{\mathrm{mr}}_{\mathsf{HE},p_m,p_k}(\mathcal{B})$	adversary $\mathcal{B}$'s advantage of recovering the correct sequence

Table 2.1: Notations and definitions for Chapter 2.

based on the number of minor alleles it has. For example, if locus i has major allele A and minor allele G, we represent AA as 0, AG (or GA) as 1, GG as 2.

Diploid Genotype and Haploid Genotype

To be consistent throughout the chapter, given a sequence of loci, we interpret an individual's **diploid genotype** as a corresponding sequence of SNVs, each of which takes values in $\{0, 1, 2\}$, and a **haploid genotype** as a corresponding sequence of alleles, each of which takes values in $\{0, 1\}$.

Linkage Disequilibrium and Recombination

Because chromosomal segments are inherited as blocks, SNVs on a sequence are usually correlated, especially when they are physically close to each other. This correlation is measured by **linkage disequilibrium** (LD) [62]. The strength of LD between two SNVs is usually represented by r^2, where $r^2 = 1$ represents the strongest LD relationship. At meiosis, two DNA sequences exchange genetic information, leading to a novel combination of alleles that is passed on to the progeny. This process is called **recombination**. The recombination rates vary on the different regions of a chromosome.

2.1.2 Honey Encryption

Honey encryption [61] is a recently proposed encryption scheme that has the advantage of providing security beyond the brute-force bound over conventional ciphers. In our case, this is a highly desirable property, considering the longevity of genomic data. Suppose a message M is sampled from a distribution p_m over the message space $\mathcal{M}$ and honey encrypted under key $K \in \mathcal{K}$ to yield a ciphertext $C \in \mathcal{C}$. Decryption under an incorrect key $K' \neq K$ yields a fake message M' also from the distribution p_m. In a conventional cipher, when decrypting a ciphertext using a wrong key, the scheme usually produces an

invalid[3] message (often denoted by special symbol $\perp$); thus the adversary can easily eliminate wrong keys via a brute-force attack. However, in honey encryption, the adversary does not have such an advantage because the output of the decryption under a wrong key is equivalent to random sampling from p_m. Honey encryption is proposed with a notion called *distribution-transforming encoder* (DTE), as we briefly describe below.

Distribution-Transforming Encoder: A DTE works by transforming the potentially non-uniform message distribution p_m into a uniform distribution over a *seed space* $\mathcal{S}$. Formally, it is a pair of algorithms represented as $\mathsf{DTE} = (\mathsf{encode}, \mathsf{decode})$: encode takes as input a message M and outputs a value in $\mathcal{S}$, whereas decode takes as input a value in $\mathcal{S}$ and outputs a message. encode is probabilistic: A message M can potentially be mapped to one of many possible values that make up a set $\mathcal{S}_\mathsf{M} \subseteq \mathcal{S}$, and $\mathcal{S}_\mathsf{M} \neq \emptyset$. For any pair of different messages M and M' (where $\mathsf{M} \neq \mathsf{M}'$), $\mathcal{S}_\mathsf{M} \cap \mathcal{S}_{\mathsf{M}'} = \emptyset$. Moreover, $\bigcup_{\mathsf{M}\in\mathcal{M}} \mathcal{S}_\mathsf{M} = \mathcal{S}$. Therefore, encode needs to choose a value randomly in $\mathcal{S}_\mathsf{M}$ when transforming M, but decode is deterministic. A good DTE has the property that a randomly selected seed, mapped to the message space, yields roughly the underlying message distribution p_m ($\frac{|\mathcal{S}_\mathsf{M}|}{|\mathcal{S}|} \approx p_m(\mathsf{M})$), where $p_m(\mathsf{M})$ is the probability of message M. We further discuss the benefits of this property in Section 2.4.

In the DTE-then-encrypt paradigm proposed in [61], encryption of a message M involves two steps: (i) application of encode to M to yield a seed s, and then (ii) encryption of s under a conventional symmetric cipher SE. HE does not provide IND-CCA (indistinguishability under chosen-ciphertext attack) security. It provides the weaker but still useful property of *message-recovery* (MR) security, described below and formally defined in Section 2.4. Consider the scenario in which an adversary wants to guess the key (K) used for the encryption. Given an ideal cipher model for SE, a randomly selected key corresponds to a permutation selected uniformly at random. Hence, if the adversary tries to decrypt a ciphertext C with a randomly guessed key K', he will obtain a value uniformly sampled from $\mathcal{S}$. If he decodes this value, the output message is equivalent to one sampled from the distribution p_m. Given a good DTE, the adversary cannot distinguish a correct key K from an incorrect one K' with a significant advantage over guessing the key (without knowledge of the ciphertext).

We use the DTE-then-encrypt construction in honey encryption. The setup is described as follows:

- Let p_m denote the distribution over the message space $\mathcal{M}$, p_k denote the distribution over the key (password) space $\mathcal{K}$, $\mathcal{S} = \{0,1\}^l$ denote the seed space with bit length l, and $\mathcal{C}$ denote the ciphertext space.
- Let $\mathsf{DTE} = (\mathsf{encode}, \mathsf{decode})$ be a DTE scheme. Specifically, $\mathsf{encode}(\mathsf{M}) = S$ and $\mathsf{decode}(S) = \mathsf{M}$, where M is a message and $S \in \mathcal{S}$.
- Use a conventional symmetric encryption scheme $\mathsf{SE} = (\mathsf{encrypt}, \mathsf{decrypt})$ with plaintext space $\mathcal{S}$ and ciphertext space $\mathcal{C}$. For block ciphers without padding, $\mathcal{C}$ is the same as $\mathcal{S}$. SE uses random bits uniformly sampled from $\{0,1\}^B$ during encryption, where B is the length of the random bits.

The honey encryption construction $\mathsf{HE}[\mathsf{DTE}, \mathsf{SE}] = (\mathsf{HEnc}, \mathsf{HDec})$ is also shown in Figure 2.1. However, as we will show, the application of HE to genomes, is far from straightfor-

[3]Here "invalid" means a message with an extremely low probability in p_m.

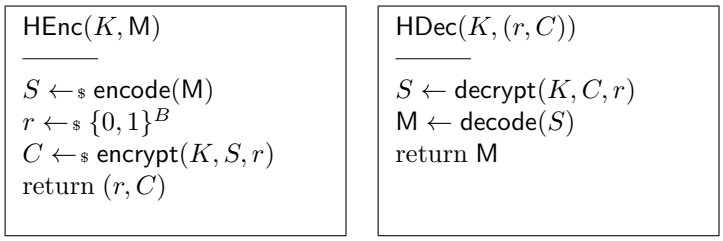

Figure 2.1: DTE-then-encrypt construction using a symmetric encryption. $\mathsf{M} \in \mathcal{M}, K \in \mathcal{K}, S \in \mathcal{S}$, and $C \in \mathcal{C}$. The symbol '\$' implies randomness of the function. r is a random salt of length B.

ward. Constructing a good DTE for genetic sequences, one that yields an HE scheme with good MR security bounds, is the main challenge addressed in this chapter. Addressing the problem of side information is also a significant challenge.

2.2 System Model

We consider a scenario where individuals' genomic data is stored in a database (e.g., a biobank) and used for various purposes, such as clinical diagnosis or therapy, or DTC services. In the data collection phase, patients provide their biological samples to a certified institution (CI) that is responsible for the sequencing. Furthermore, each patient also chooses a password (we assume patients can choose low-entropy passwords). The CI pre-processes the sequence data; the most important step is the application of protection mechanisms to the data, such as encryption using the passwords of the patients. The CI then sends the processed data to the biobank. To efficiently protect the data, we assume there are two layers of protection:

- The inner-layer protection is provided by using cryptographic techniques. This layer is necessary for defending against attacks from insiders or someone who hacks into the system and steals the database. This is the focus of this chapter.
- The outer-layer protection is the access control; it decides various permissions on the data. Access control has been extensively investigated in the literature [63] and is out of the scope of this chapter.

During data retrieval, a user (such as a doctor or the patient himself) first authenticates himself to the system using a passcode[4], or biometric information (e.g., face). After authentication, the user can send a data request to the biobank that processes the request according to access control rules and the biobank responds with the authorized data. Figure 2.2 gives an overview of the considered architecture.

2.2.1 Genomic Data Representation

We represent each patient's genomic data as a sequence of genetic variants (SNVs) that take values from the set $\{0, 1, 2\}$, as we discussed before. We assume a sequence M with

[4]Chosen by the user or generated by a one-time passcode generator. Note that the passcode used for authentication cannot be the same as the password used for PBE (if PBE is used in GenoGuard that is introduced in Section 2.3), as the former would require storing a hash of the passcode on the system.

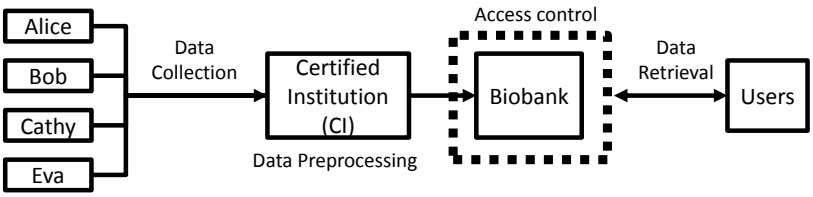

Figure 2.2: System model of genomic data storage and retrieval. Patients provide their samples to CI for sequencing. Encrypted sequence data is sent to the biobank and retrieved for various purposes by the users.

n SNVs, and we represent such a sequence as $(m_1, m_2, \cdots, m_n)$, where m_i represents an SNV. We use $\mathsf{M}_{i,j}$ to represent the subsequence including all the SNVs between (and including) the i-th and the j-th.

2.2.2 Threat Model

We assume the CI to be trusted in order to perform sequencing on patients' samples. An adversary can be anyone (except the CI) who has access to the protected data, such as the biobank, a user who has been granted access permission on part of the data, or an attacker who breaks into the biobank and downloads a snapshot of the database. As a consequence, the adversary can be assumed to have a copy of encrypted sequences. We further assume that the adversary has access to public knowledge about genomics, i.e., AF, LD, recombination and mutation rates. A stronger adversary could even have some side information about a given patient, such as his phenotype, and even some of his SNVs. We represent the adversary's *background knowledge* as $\mathsf{BK} = \{\text{AF}, \text{LD}, \text{recombination and mutation rates}, [\text{side info}]\}$, where "[side info]" means the type and amount of side information depend on the power of an adversary. We also study the effect of phenotype as side information (in Section 2.5) and propose a general solution in this regard. We emphasize that more side information could result in stronger attacks. Throughout this chapter, we assume a *computationally unbounded adversary* who has the capability to efficiently enumerate all keys in $\mathcal{K}$ and to use them to decrypt the data, also called a *brute-force attack*. We also assume that the adversary is honest-but-curious (i.e., follows the protocols honestly, but tries to learn more information than he is authorized for). The adversary's main goal is to break the inner-layer protection and gain access to the plaintext sequences of the patients.

2.3 GenoGuard

We describe *GenoGuard*, our solution based on honey encryption, for the secure storage of genomic data. We show the main steps of the protocol in Figure 2.3. We represent the patient and the user as two separate entities, but they can be the same individual, depending on the application. We discuss more about the application scenarios in Section Section 2.6. Step by step, we discuss the protocol in this section, emphasizing the encoding (Step 3) and decoding (Step 9) steps that are the major features of GenoGuard.

Initially, a patient provides his biological sample (e.g., blood or saliva) to the CI and chooses a password that is used for the encryption (Step 1). The CI does the

sequencing on the sample and produces genomic data represented as discussed in Section 2.2.1 (Step 2).

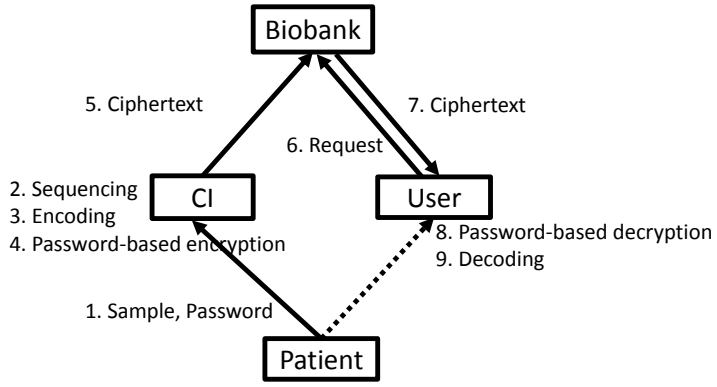

Figure 2.3: GenoGuard protocol. A patient provides his biological sample to the CI, and chooses a password for honey encryption. The CI does the sequencing, encoding and password-based encryption, and then sends the ciphertext to the biobank. During a retrieval, a user (e.g., the patient or his doctor) requests for the ciphertext, decrypts it and finally decodes it to get the original sequence.

2.3.1 Encoding

We introduce a novel DTE scheme that can be applied efficiently on genome sequences. The general idea is to estimate the conditional probability of an SNV given all preceding ones. In other words, the proposed scheme estimates $P(m_i|\mathsf{M}_{1,i-1})$, the conditional probability of the i-th SNV given preceding SNVs. The probability of a complete sequence M can be decomposed as follows:

$$p_m(\mathsf{M}) = P(m_n|\mathsf{M}_{1,n-1})P(m_{n-1}|\mathsf{M}_{1,n-2})\cdots P(m_2|m_1)P(m_1). \tag{2.1}$$

The main challenge is to find an efficient way to encode a sequence M into a uniformly distributed seed, which defines the deterministic mapping from M to $\mathcal{S}_\mathsf{M}$ (then we can uniformly pick a value from $\mathcal{S}_\mathsf{M}$). A naive and impractical method would be to enumerate all possible sequences, compute their corresponding probabilities, calculate the cumulative distribution function (CDF) of each sequence in a pre-defined order, and finally assign the corresponding portion of seeds to a sequence. However, given that there are three possible states for each SNV on a sequence of length n, this method incurs both time and space complexity of $O(3^n)$.

Therefore, we propose a novel approach for efficiently encoding such a sequence. The approach works by assigning subspaces of $\mathcal{S}$ to the prefixes of a sequence M. The prefixes of a sequence M are all the subsequences in the set $\{\mathsf{M}_{1,i}|1 \le i \le n\}$. For example, the prefixes of the sequence ATTCG are $\{\mathsf{A}, \mathsf{AT}, \mathsf{ATT}, \mathsf{ATTC}, \mathsf{ATTCG}\}$. We first describe the basic setup as follows:

- Seed space $\mathcal{S}$ corresponds to the interval $[0, 1)$. Each seed is a real number in this interval. In practice, we need to use only sufficient precision (l bits as indicated by the definition $\mathcal{S} = \{0, 1\}^l$) to distinguish between the seeds of different sequences.

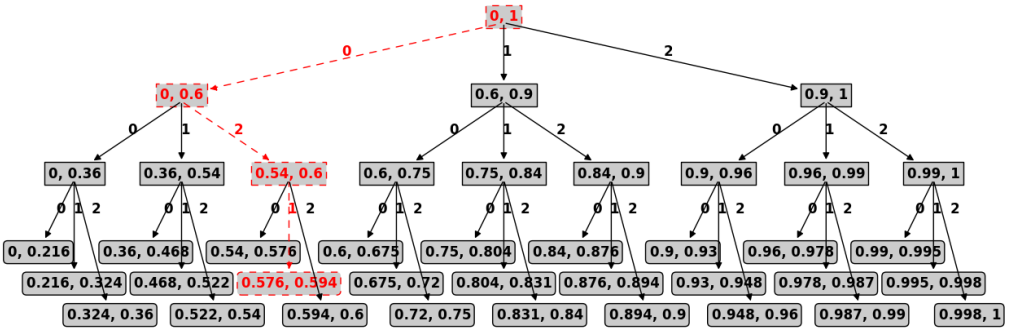

Figure 2.4: A toy example of the encoding process. The sequence is of length 3. The sequence that needs to be encoded is $(0, 2, 1)$, shown in red dashed line. Take the second step as an example. We have $P(m_2 = 0|m_1 = 0) = 0.6, P(m_2 = 1|m_1 = 0) = 0.3, P(m_2 = 2|m_1 = 0) = 0.1$, and $[L_1^0, U_1^0) = [0, 0.6)$. Hence the next three intervals are: **(i)** $[L_2^0, U_2^0) = [L_1^0, L_1^0 + (U_1^0 - L_1^0) \times P(m_2 = 0|m_1 = 0)) = [0, 0.36)$; **(ii)** $[L_2^1, U_2^1) = [L_1^0 + (U_1^0 - L_1^0) \times P(m_2 = 0|m_1 = 0), L_1^0 + (U_1^0 - L_1^0) \times (P(m_2 = 0|m_1 = 0) + P(m_2 = 1|m_1 = 0))) = [0.36, 0.54)$; **(iii)** $[L_2^2, U_2^2) = [L_1^0 + (U_1^0 - L_1^0) \times (P(m_2 = 0|m_1 = 0) + P(m_2 = 1|m_1 = 0)), U_1^0) = [0.54, 0.6)$. Note that the intervals in black solid line do not need to be computed when encoding $(0, 2, 1)$. When we reach the leaf $[0.576, 0.594]$, we pick a seed randomly from this range, e.g., 0.583.

But, for simplicity of presentation in the rest of this subsection, we assume there is infinite precision.

- To calculate the CDFs, we define a total order $\mathcal{O}$ of all sequences in $\mathcal{M}$, i.e., $\mathcal{O} : \mathcal{M} \to \mathbb{N}$. For any two different sequences M and M', scanning from the first SNV, suppose they begin to differ at the i-th SNV, m_i and m_i' correspondingly (i.e., $\mathsf{M}_{1,i-1} = \mathsf{M}'_{1,i-1}$ and $m_i \neq m_i'$). If the value (0, 1, or 2) of m_i is smaller than that of m_i', then $\mathcal{O}(\mathsf{M}) < \mathcal{O}(\mathsf{M}')$, otherwise $\mathcal{O}(\mathsf{M}) > \mathcal{O}(\mathsf{M}')$. The CDF of a sequence M is $\mathsf{CDF}(\mathsf{M}) = \sum_{\substack{\mathsf{M}' \in \mathcal{M} \\ \mathcal{O}(\mathsf{M}') \leq \mathcal{O}(\mathsf{M})}} p_m(\mathsf{M}')$ where $p_m(\mathsf{M}')$ is the probability of sequence M'.

In a nutshell, we can encode a sequence with the help of a perfect ternary tree (an example of which is provided in Figure 2.4). For a sequence M, starting from the root, (i) if an SNV m_i is 0, we move down to the left branch; (ii) if it is 1, we move down to the middle branch; (iii) if it is 2, we move down to the right branch. As a consequence, each internal node represents a prefix of a sequence, whereas each leaf node represents a complete sequence. We also attach an interval $[L_i^j, U_i^j)$ to each node, where i represents the depth of the node in the tree, and j represents the order of the node at a given depth i, both starting from 0. This interval is the sub seed space that can be assigned to the sequences that start with the prefix represented by the corresponding node.

Here, we describe the details of encoding process (step 3 in Figure 2.3). Assume we encode a sequence M. It is obvious that the root has an interval $[0, 1)$, namely, $[L_0^0, U_0^0) = [0, 1)$. Depending on the value of SNV m_{i+1}, encoding proceeds from the node that represents $\mathsf{M}_{1,i}$ with order j at depth i to depth $i + 1$ as follows:

- If $m_{i+1} = 0$, go to the left branch and attach an interval $[L_{i+1}^{3j}, U_{i+1}^{3j}) = [L_i^j, L_i^j + (U_i^j - L_i^j) \times P(m_{i+1} = 0|\mathsf{M}_{1,i}))$.

- If $m_{i+1} = 1$, go to the middle branch and attach an interval $[L_{i+1}^{3j+1}, U_{i+1}^{3j+1}) = [L_i^j + (U_i^j - L_i^j) \times P(m_{i+1} = 0|\mathsf{M}_{1,i}), L_i^j + (U_i^j - L_i^j) \times (P(m_{i+1} = 0|\mathsf{M}_{1,i}) + P(m_{i+1} = 1|\mathsf{M}_{1,i})))$.
- If $m_{i+1} = 2$, go to the right branch and attach an interval $[L_{i+1}^{3j+2}, U_{i+1}^{3j+2}) = [L_i^j + (U_i^j - L_i^j) \times (P(m_{i+1} = 0|\mathsf{M}_{1,i}) + P(m_{i+1} = 1|\mathsf{M}_{1,i})), U_i^j)$.

So far, we have not devoted much content to the discussion of computing the conditional probability $P(m_{i+1}|\mathsf{M}_{1,i})$, which will be elaborated later. For now, we focus on how the encoding scheme works on the high level. Finally, when we reach the leaf node with the interval $[L_n^j, U_n^j)$, we pick a seed S uniformly from this range to encode the corresponding sequence. In the following, we give a toy example of this encoding process.

Example (Encoding): Suppose all sequences are of length 3. The sequence M that needs to be encoded is $(0, 2, 1)$. Assume $P(m_1 = 0) = 0.6, P(m_2 = 2|m_1 = 0) = 0.1$, and $P(m_3 = 1|\mathsf{M}_{1,2}) = 0.3$. The encoding process is illustrated in Figure 2.4.

In Step 4 (in Figure 2.3), after the encoding is finished, the seed, as a plaintext, is fed into a conventional password-based encryption (PBE) [64] by using the password chosen by the patient (at Step 1). This step is a direct application of PBE, so we skip the details here. The encrypted seed is then sent to the biobank (step 5) that, as a centralized database, receives requests (step 6) from users and responds with the corresponding encrypted data (step 7).

2.3.2 Decoding

When an encrypted seed is sent to the user, the user first performs a password-based decryption by using the patient's password (step 8). As discussed, the user could be the patient himself, or the patient can provide his password on behalf of the user. We discuss more on these scenarios in Section 2.6. Once the user has the plaintext seed, the decoding process (step 9) is the same as the encoding process. Given a seed $S \in [0, 1)$, at each step, the algorithm computes three intervals for the three branches, chooses the interval in which the seed S falls, and goes down along the ternary tree. Once it reaches a leaf node, it outputs the path from the root to this leaf with all chosen SNVs.

2.3.3 Moving to Finite Precision

As we mentioned, the current seed space $\mathcal{S}$ is a real number domain with infinite precision. However, considering the size of a DNA sequence, with infinite precision, we could end up having a very long floating-point representation for a sequence, which could cause a high storage overhead. Also, we cannot afford to enumerate all possible sequences to find the smallest precision to represent all the corresponding real numbers. Moreover, if we work with finite precision and decide on the precision *a priori* (without enumerating the sequences), this could result in an inaccurate representation of the sequence distribution, thus causing a security loss. In this subsection, we describe how our proposed DTE scheme can be implemented with finite precision and with negligible effect on security.

For a sequence of length n, with each SNV taking three possible values, we require at least $(n \cdot \log_2 3)$ bits to store the sequence.[5] To optimally implement the scheme, we

[5] We do not consider compression techniques here.

first select a storage overhead parameter h ($h > \log_2 3$). We use hn bits to encode one sequence. As before, the algorithm works by segmenting intervals based on conditional probabilities. In this case, however, an interval is represented by integers, and not by real numbers of infinite precision. The root interval is $[0, 2^{hn} - 1]$. To better describe the scheme, suppose (during the encoding) we reach the j-th node at depth i on the tree (the root has depth 0 and the leaves have depth n). The interval of this node is denoted by $[L_i^j, U_i^j]$ (U_i^j inclusive, which is different from the infinite-precision case). The segmentation rules are described in the following.

We compute the conditional probabilities for the three branches, P_L (left branch), P_C (middle branch) and P_R (right branch) respectively. Without loss of generality, we assume the three probabilities are ordered as $P_L \geq P_C \geq P_R$ (the following algorithm is similar for different orderings). We initialize a variable $\mathbf{avail} = U_i^j - L_i^j + 1$ to denote the size of the seed space available for allocation. The sizes of seed space that will be allocated to the three branches are denoted by $\mathbf{alloc}_L$ (left branch), $\mathbf{alloc}_C$ (middle branch), and $\mathbf{alloc}_R$ (right branch). Note that $\mathbf{alloc}_L + \mathbf{alloc}_C + \mathbf{alloc}_R = U_i^j - L_i^j + 1$. The algorithm advances as follows:

(i). If $P_R < \frac{3^{n-i-1}}{\mathbf{avail}}$, then $\mathbf{alloc}_R = 3^{n-i-1}$, otherwise $\mathbf{alloc}_R = \lceil P_R \cdot \mathbf{avail} \rceil$. Then, we update $\mathbf{avail}$ as $\mathbf{avail} = \mathbf{avail} - \mathbf{alloc}_R$.

(ii). If $\frac{P_C}{P_C + P_L} < \frac{3^{n-i-1}}{\mathbf{avail}}$, then $\mathbf{alloc}_C = 3^{n-i-1}$, otherwise $\mathbf{alloc}_C = \lceil P_C \cdot \mathbf{avail} \rceil$. And, we set $\mathbf{alloc}_L = \mathbf{avail} - \mathbf{alloc}_C$.

(iii). Finally, we set the three sub-intervals as:

- $[L_{i+1}^{3j}, U_{i+1}^{3j}] = [L_i^j, L_i^j + \mathbf{alloc}_L - 1]$;
- $[L_{i+1}^{3j+1}, U_{i+1}^{3j+1}] = [L_i^j + \mathbf{alloc}_L, L_i^j + \mathbf{alloc}_L + \mathbf{alloc}_C - 1]$;
- $[L_{i+1}^{3j+2}, U_{i+1}^{3j+2}] = [L_i^j + \mathbf{alloc}_L + \mathbf{alloc}_C, U_i^j]$.

The intuition behind the above conditions is that we need to allocate at least one integer (seed) for one sequence. To ensure this, when we want to move down to a branch, we need to guarantee that the size of the seed space allocated for this branch is not smaller than the total number of sequences belonging to this branch. The requirement is satisfied from the beginning by setting the root interval as $[0, 2^{hn} - 1]$ and never violated in the algorithm. This method causes a deviation from the original sequence distribution. In Section 2.4, we quantify the security loss due to such deviation and prove that it is negligible.

2.3.4 Modeling Genome Sequences

To compute the conditional probabilities in Equation (2.1) efficiently, we introduce several models and compare their goodness of fit in real genome datasets.

Modeling with linkage disequilibrium and allele frequency

With LD and AF, we can compute the joint probability of two SNVs, $P(m_i, m_j)$. However, to compute the conditional probability $P(m_{i+1}|\mathsf{M}_{1,i})$, we have to simplify the model (Equation (2.1)) because public LD values are always given pairwise in the literature.

Although there could be multiple pairwise LD relations for SNV_{i+1}, we adopt the following heuristic method: We consider only the previous SNV that has the strongest LD with SNV_{i+1}. Such an LD usually occurs between neighboring SNVs on the DNA sequence, hence we have $P(m_{i+1}|\mathsf{M}_{1,i}) \approx P(m_{i+1}|m_i) = \frac{P(m_{i+1},m_i)}{P(m_i)}$. This is the first-order Markov chain that was considered also in genomics [65].

This model fails to capture the correlation between distant SNVs. However, we argue that it approximates the genome sequence model better than the uniform distribution model used in conventional encryption, as we will see later in model comparison with real datasets.

Modeling by building k-th-order Markov chains on a dataset

With this method, we assume the correlation in a genome sequence can be captured by a k-th-order Markov chain, where the conditional probability of SNV_{i+1} depends on the k preceding SNVs. In other words, we estimate the conditional probability as

$$P(m_{i+1}|\mathsf{M}_{1,i}) \approx P(m_{i+1}|\mathsf{M}_{i-k+1,i}). \tag{2.2}$$

Researchers have tried to build such a genetic Markov model in a different context [66]. However, to the best of our knowledge, there is no public data (like LD) available for these models. In a similar manner, we build the k-th-order Markov model on a real dataset, for different k values. Assume the dataset has N sequences. We use $F(\mathsf{M}_{i,j})$ to represent the frequency of subsequence $\mathsf{M}_{i,j}$ between SNVs i and j in the dataset. The k-th-order Markov model is built by computing

$$P(m_{i+1}|\mathsf{M}_{i-k+1,i}) = \begin{cases} 0 & \text{if } F(\mathsf{M}_{i-k+1,i}) = 0, \\ \frac{F(\mathsf{M}_{i-k+1,i+1})}{F(\mathsf{M}_{i-k+1,i})} & \text{if } F(\mathsf{M}_{i-k+1,i}) > 0. \end{cases} \tag{2.3}$$

Due to the constraint of the dataset size, k normally can only take small values to avoid overfitting of the model. For example, in HapMap diploid genotype datasets, N is smaller than 200 for each population. For $k = 3$, there are 81 possible configurations for $\mathsf{M}_{i-k+1,i+1}$, which makes the average frequency for each configuration quite small, hence the model has modest statistical significance due to this sparsity problem. We introduce this model as a possible direction and use it to emphasize the importance of higher-order correlation, which will be shown in the evaluation. The k-th-order Markov chain serves as a bridge to the next more promising model.

Modeling with recombination rates

Although higher-order Markov models might better model genome sequences, these models seem unlikely to be practical because of the difficulty of accurately estimating all the necessary parameters in available datasets. Inspired by the modeling method used by Li and Stephens [67], we can address the problem from a different viewpoint. Given a set of k existing haploid genotypes $\{h_1, h_2, ..., h_k\}$, another haploid genotype h_{k+1} to be observed is an imperfect mosaic of $h_1, h_2, ..., h_k$, due to genetic recombination and mutation (Figure 2.5). This reproduction process is actually a hidden Markov model with a sequence of n states (the number of loci in a haploid genotype):

- **Markov chain states:** State j, X_j, can take a value from 1 to k, representing the original haploid genotype for locus j;

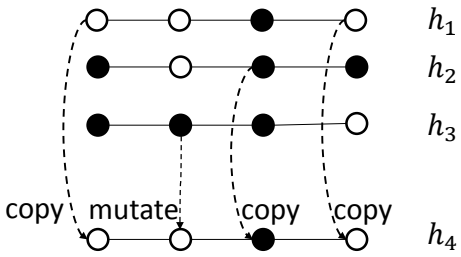

Figure 2.5: An example showing how the haploid genotype h_4 is interpreted as an imperfect mosaic of a given set of haploid genotypes $\{h_1, h_2, h_3\}$, based on recombination and mutation. Each haploid genotype can be as long as the whole genome, but we show only four loci here to explain the idea. White circle means allele 0 for that locus, whereas black circle means allele 1. The first allele of haploid genotype h_4 is copied from h_1. Though the second allele comes from h_3, it mutates to a different allele. The third allele is copied from h_2, and the fourth is copied from h_1. Note that this shows just one possible process to get h_4 from $\{h_1, h_2, h_3\}$, and as there are many other possibilities, the task of this model is to compute the probability of observing h_4 by taking all the possible underlying processes into account, which constitutes a hidden Markov model.

- **Symbol emission probabilities:** $h_{i,j}$ denotes the allele (0 or 1) at locus j in haploid genotype i. To produce h_{k+1}, at state j, an allele $h_{k+1,j}$ is output with a certain probability, depending on the allele of the original haploid genotype (X_j) and the mutation rate;

- **Transition probabilities:** Transition probabilities from state j to state $j+1$ depend on the recombination rate between locus j and $j+1$.

With this model, we can compute the probability of a haploid genotype h_{k+1}, that is, $P(h_{k+1}|h_1, ..., h_k)$. The computation is done with the well-known forward-backward algorithm for hidden Markov models [68]. The probability of a genome sequence M, which is the coupling of two haploid genotypes, can be computed similarly by extending this hidden Markov model so that state j will take a value pair (X_j^1, X_j^2), where X_j^1 denotes the first original haploid genotype and X_j^2 denotes the second. Such an extension technique has been detailed in a genotype imputation scenario [69]. The conditional probability $P(m_{i+1}|\mathsf{M}_{1,i})$ can then be computed in the intermediate steps of the forward algorithm. Details are specified as follows.

Initially, at state 1, we have $P(X_1 = x) = \frac{1}{k}(x \in \{1, \cdots, k\})$. The transition probability from state j to $j+1$ is characterized by

$$\begin{aligned} & P(X_{j+1} = x' | X_j = x) \\ = & \begin{cases} exp(-\frac{\rho_j}{k}) + \frac{1-exp(-\frac{\rho_j}{k})}{k} & \text{if } x' = x; \\ \frac{1-exp(-\frac{\rho_j}{k})}{k} & \text{otherwise,} \end{cases} \end{aligned} \tag{2.4}$$

where ρ_j is the genetic distance between locus j and $j+1$. It is computed based on the recombination rate between these two loci. Intuitively, a smaller genetic distance will make the two states more likely to take the same value, meaning that they are more likely to come from the same haploid genotype.

At state j, an allele (0 or 1) will be emitted. To mimic the effects of mutation, the emitting probability is characterized by

$$P(h_{k+1,j} = a|X_j = x) = \begin{cases} 1-\lambda & \text{if } h_{x,j} = a; \\ \lambda & \text{otherwise,} \end{cases} \tag{2.5}$$

where a is 0 or 1, and λ is the mutation rate.

Let the forward variable $\alpha_j(x) = P(h_{k+1,\leq j}, X_j = x)$. Then $\alpha_1(x) = P(h_{k+1,1}|X_1 = x)P(X_1 = x)$. And $\alpha_2(x), \cdots, \alpha_n(x)$ can be computed recursively using

$$\begin{aligned} \alpha_{j+1}(x) =& P(h_{k+1,j+1}|X_{j+1} = x) \\ & \sum_{x'=1}^{k} \alpha_j(x')P(X_{j+1} = x|X_j = x'), \end{aligned} \tag{2.6}$$

The probability of a complete haploid genotype is then computed using

$$P(h_{k+1}|h_1, \cdots, h_k) = \sum_{x=1}^{k} \alpha_n(x). \tag{2.7}$$

The conditional probability for allele $j+1$, given all preceding alleles, is computed using

$$P(h_{k+1,j+1}|h_{k+1,\leq j}, h_1, \cdots, h_k) = \frac{\sum_{x=1}^{k} \alpha_{j+1}(x)}{\sum_{x=1}^{k} \alpha_j(x)}. \tag{2.8}$$

For a genome sequence that couples two haploid genotypes, all the above quantities can be computed similarly by an extension of this hidden Markov model [69].

The correlation between two SNVs, which is considered in the previous two models, is essentially the result of recombination in genome sequences. With this recombination model, we are able to capture the high-order correlation efficiently, without having to estimate a large number of parameters.

Goodness of fit of the models

To evaluate the models, we used different types of real genomic datasets from HapMap, for the population CEU (Utah residents with Northern and Western European ancestry from the CEPH collection) [70], including:

- A diploid genotype dataset that contains 165 individuals, each having 22 pairs of autosomes (different from sex chromosomes that are discussed in Section 2.5.1). The shortest chromosome contains 17304 SNVs, whereas the longest one contains 102157 SNVs;
- A haploid genotype dataset that contains 234 haploid genotypes, each of which has the same sequence of loci as that in the diploid genotype dataset on the 22 chromosomes;
- Allele frequency and linkage disequilibrium datasets for each chromosome;
- Recombination rates for each chromosome.

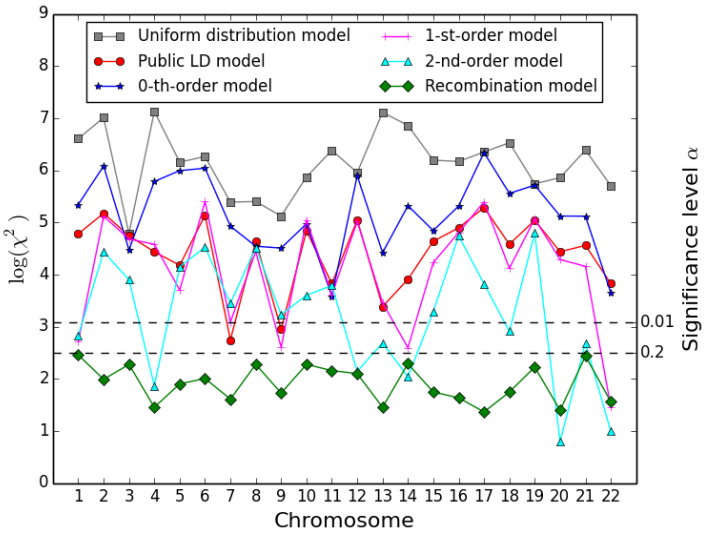

Figure 2.6: Chi-square goodness-of-fit tests for different genome sequence models on 22 chromosomes. The x-axis is the chromosome number, from 1 to 22. To graphically show the results at a fine scale, the left y-axis is transformed to the logarithm of chi-squared statistic. The right y-axis shows one frequently used significance level, $\alpha = 0.01$, and another significance level, $\alpha = 0.2$. The uniform distribution model is the one used in conventional encryption. The "public LD model" is built with public LD and AF data. The "0-th", "1-st", "2-nd"-order models are the Markov models built on the dataset. Finally, the "recombination model" is built based on genetic recombination and mutation. Most models are rejected at $\alpha = 0.01$, whereas the recombination model cannot be rejected even at $\alpha = 0.2$, which shows a good fit of this model on real datasets.

We performed a chi-square goodness-of-fit test to show how well each model fits the diploid genotype dataset. We divided the sequence space $\mathcal{M}$ into B bins with equal probability. The chi-square statistic is defined as

$$\chi^2 = \sum_{i=1}^{B} \frac{(O_i - E_i)^2}{E_i}, \tag{2.9}$$

where O_i is the observed frequency for bin i, and E_i is the expected frequency for bin i. The null hypothesis H_0 is that the data follows the specified distribution model. B is chosen with an empirical formula in statistical theory [71] ($B = \lfloor 1.88N^{\frac{2}{5}} \rfloor$ where N is the sample size). We performed several rounds of the test for different B values around the empirical one and they all gave similar results. Hence we set B to be 10, and show the results in Figure 2.6. From the chi-square statistics, we can see that uniform distribution indeed gives a poor model of genome sequences. The 0-th-order model built on the dataset is also not appropriate because it does not take the correlation among SNVs into account. The model built with public LD and AF performs similarly with the first-order model built on the dataset, which is reasonable because they both consider only the first-order correlation. The second-order model is better than the previous four models, but it is not stable across different chromosomes: in many chromosomes, we can reject the null hypothesis H_0 at the significance level (α) of 0.01. The recombination model performs best among these models because it captures high-order correlations that are

naturally caused by the underlying recombination mechanism. Moreover, the model is stable across all tests and cannot be rejected, even at the significance level of 0.2 in every chromosome, which shows a good fit of this model on real datasets. Therefore, we keep this model for our scheme.

2.4 Security Analysis

In this section, we prove the security of our proposed DTE scheme, with regard to the scheme in finite precision.

Once the algorithm allocates seed space of size 3^{n-i-1} to a branch at step i (as in Section 2.3.3), each following step simply segments an input interval into three parts of equal size. Hence there is only one seed for each sequence in the sub-tree under the branch of step i. As discussed in Section 2.3.3, in such a case, the subinterval of the j^{th} node at depth i of the tree will contain 3^{n-i-1} integers that are exactly the number of sequences under that branch.

The goal in constructing a DTE is that decode applied to uniform points (in the seed space) provides sampling close to that of the target distribution p_m; this is the sequence distribution produced by the k^{th}-order Markov chain. The seed space $\mathcal{S}$ is the integer interval $[0, 2^{hn} - 1]$ (i.e., $l = hn$). We define p_d to be the DTE message distribution over $\mathcal{M}$ by

$$p_d(\mathsf{M}) = P[\mathsf{M}' = \mathsf{M} : S \leftarrow_{\$} \mathcal{S}; \mathsf{M}' \leftarrow \mathsf{decode}(S)].$$

The additional security provided by honey encryption depends on the difference between p_m and p_d. Intuitively, p_m and p_d are "close" in a secure DTE. Next, we quantify this difference for the proposed DTE scheme. Let P_m^i be the original probability of the prefix sequence $\mathsf{M}_{1,i}$, namely, $P_m^i = \sum_{\substack{\mathsf{M}' \in \mathcal{M} \\ \mathsf{M}'_{1,i} = \mathsf{M}_{1,i}}} p_m(\mathsf{M}')$. We define P_d^i similarly in the distribution p_d.

We prove the following lemma that gives an upper bound on the difference between the original message distribution p_m and the DTE message distribution p_d.

Lemma 2.1. $\forall \mathsf{M} \in \mathcal{M}, |p_m(\mathsf{M}) - p_d(\mathsf{M})| < \frac{1}{2^{(h - \log_2 3)n}}$.

Proof. As we showed in the optimized scheme, the goal is to compute three intervals according to the conditional probabilities for the three branches, P_L (left branch), P_C (middle branch) and P_R (right branch), respectively. Without loss of generality, we assume the three probabilities are ordered as $P_L \geq P_C \geq P_R$. The proof is similar for different orderings.

First consider the case when $p_d(\mathsf{M}) > p_m(\mathsf{M})$. According to the algorithm, the probability of sequence M increases in p_d only if it has at least one SNV that belongs to the right branch or the middle branch. In other words, when we set $\mathbf{alloc}_R = 3^{n-i-1}$ ($\mathbf{alloc}_C = 3^{n-i-1}$) or $\mathbf{alloc}_R = \lceil P_R \cdot \mathbf{avail} \rceil$ ($\mathbf{alloc}_C = \lceil P_C \cdot \mathbf{avail} \rceil$), we actually increase the probability for this branch. Without loss of generality, we prove for the right branch in the following, but it is similar for the middle branch.

If $\mathbf{alloc}_R = 3^{n-i-1}$ has been executed at some step, as we mentioned, there will be only one integer assigned for each sequence under that branch. The probability of one integer in $\mathcal{S}$ is $\frac{1}{2^{hn}}$. Hence, the probability of sequence M will be increased by at most $\frac{1}{2^{hn}}$.

Otherwise, the increased probability is only due to $\mathbf{alloc}_R = \lceil P_R \cdot \mathbf{avail} \rceil$, coming from the "ceiling" operation that expands the interval by at most one additional integer. In this case, we need to show that

$$\forall i \in \{0, 1, 2, \cdots, n\}, P_d^i - P_m^i \leq \frac{i}{2^{hn}}. \tag{2.10}$$

When $i = 0$, there is no prefix and $P_d^0 = P_m^0 = 1$, and hence the result holds. Also, for $i <= k$, the result holds. When $i = k + 1$, $P_m^{k+1} = P_m^k \cdot P_R$. The right sub-interval has size $\mathbf{alloc}_R = \lceil P_R \cdot (U_k^j - L_k^j + 1) \rceil$. Then we have

$$\begin{aligned} P_d^{k+1} &= \frac{\lceil P_R \cdot (U_k^j - L_k^j + 1) \rceil}{2^{hn}} \leq \frac{P_R \cdot (U_k^j - L_k^j + 1) + 1}{2^{hn}} \\ &= P_d^k \cdot P_R + \frac{1}{2^{hn}}. \end{aligned} \tag{2.11}$$

Hence, $\forall i >= 0, 2^i > i$

$$\begin{aligned} P_d^{k+1} - P_m^{k+1} &= P_d^k \cdot P_R - P_m^k \cdot P_R + \frac{1}{2^{hn}} \\ &\leq \frac{k}{2^{hn}} \cdot P_R + \frac{1}{2^{hn}} < \frac{k+1}{2^{hn}}. \end{aligned} \tag{2.12}$$

Therefore, we have $p_d(\mathsf{M}) - p_m(\mathsf{M}) = P_d^n - P_m^n \leq \frac{n}{2^{hn}} < \frac{2^n}{2^{hn}} = \frac{1}{2^{(h - \log_2 3)n}}$.

If $p_d(\mathsf{M}) < p_m(\mathsf{M})$, we need to show that $p_d(\mathsf{M}) - p_m(\mathsf{M}) > -\frac{1}{2^{(h-1)n}}$. Consider the smallest depth i which makes $P_d^i < P_m^i$. Let $P_d^i = P_m^i - \epsilon_0$, where $\epsilon_0 \geq 0$. In the previous step, the algorithm must have chosen the left branch (or the middle branch), otherwise P_d^i will continue to be larger than P_m^i. We only prove for the left branch whose probability will decrease the most.[6] Then, we have

$$\begin{aligned} P_d^i &\geq P_d^{i-1} \cdot P_L - \frac{3^{n-i} \cdot 2}{2^{hn}} \\ &\geq P_m^{i-1} \cdot P_L - \frac{3^{n-i} \cdot 2}{2^{hn}} = P_m^i - \frac{3^{n-i} \cdot 2}{2^{hn}}, \end{aligned} \tag{2.13}$$

and hence $\epsilon_0 < \frac{3^{n-i} \cdot 2}{2^{hn}}$.

We define $P_d^{i+k} = P_m^{i+k} - \epsilon_k$. Now, we will show that $\epsilon_k < \sum_{j=0}^{k} \frac{3^{n-i-j} \cdot 2}{2^{hn}}$. Clearly, the result holds for $k = 0$. Going from depth $i + k$ to $i + k + 1$, we have

$$\begin{aligned} P_d^{i+k+1} &\geq P_d^{i+k} \cdot P_L - \frac{3^{n-i-k-1} \cdot 2}{2^{hn}} \\ &> (P_m^{i+k} - \epsilon_k) \cdot P_L - \frac{3^{n-i-k-1} \cdot 2}{2^{hn}} \\ &> P_m^{i+k+1} - \sum_{j=0}^{k+1} \frac{3^{n-i-j} \cdot 2}{2^{hn}}. \end{aligned} \tag{2.14}$$

Hence,

$$\begin{aligned} p_d(\mathsf{M}) - p_m(\mathsf{M}) = -\epsilon_{n-i} &> -\sum_{j=0}^{n-i} \frac{3^{n-i-j} \cdot 2}{2^{hn}} \\ &> -\sum_{j=0}^{n-1} \frac{3^{n-1-j} \cdot 2}{2^{hn}} > -\frac{1}{2^{(h - \log_2 3)n}}. \end{aligned} \tag{2.15}$$

[6] The result also holds for the middle branch whose probability can decrease (or increase), but not as much as the left branch (or the right branch).

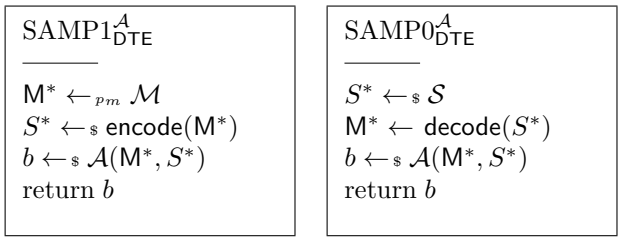

Figure 2.7: Game defining the DTE advantage. In SAMP1$^{\mathcal{A}}_{\mathsf{DTE}}$, sequence M^* is sampled according to p_m, whereas in SAMP0$^{\mathcal{A}}_{\mathsf{DTE}}$, M^* is equivalently sampled according to p_d. The adversary's output b is 0 or 1, indicating his guess on whether he is in SAMP0$^{\mathcal{A}}_{\mathsf{DTE}}$ or SAMP1$^{\mathcal{A}}_{\mathsf{DTE}}$.

The above proof shows that, for a given sequence M with original probability $p_m(\mathsf{M})$, the corresponding probability $p_d(\mathsf{M})$ in the DTE model (Section 2.3.3) can increase or decrease at most by $\frac{1}{2^{(h-\log_2 3)n}}$, which is useful to prove a negligible adversary advantage in later theorems (Theorem 2.2 and Theorem 2.1). ■

Lemma 2.1 bounds the largest difference between $p_m(\mathsf{M})$ and $p_d(\mathsf{M})$. It gives rise to the following important theorem that bounds the DTE advantage of an adversary, introduced by honey encryption. The DTE advantage is formally defined by the following definition.

Definition 2.1. *Let $\mathcal{A}$ be an adversary attempting to distinguish between the two games shown in Figure 2.7. The advantage of $\mathcal{A}$ for the sequence distribution p_m and encoding scheme $\mathsf{DTE} = (\mathsf{encode}, \mathsf{decode})$ is*

$$\mathrm{Adv}^{\mathrm{dte}}_{\mathsf{DTE},p_m}(\mathcal{A}) = |P[\mathrm{SAMP1}^{\mathcal{A}}_{\mathsf{DTE}} \Rightarrow 1] - P[\mathrm{SAMP0}^{\mathcal{A}}_{\mathsf{DTE}} \Rightarrow 1]|.$$

Theorem 2.1. *Let p_m be the sequence distribution and $\mathsf{DTE} = (\mathsf{encode}, \mathsf{decode})$ be the transformation scheme using hn bits. Let $\mathcal{A}$ be any sampling adversary, then*

$$\mathrm{Adv}^{\mathrm{dte}}_{\mathsf{DTE},p_m}(\mathcal{A}) \leq \frac{1}{2^{(h-2\log_2 3)n}}.$$

Proof. The proof follows Theorem 6 in [61]. We briefly describe it in the following.

$$\begin{aligned}
&P[\mathrm{SAMP1}^{\mathcal{A}} \Rightarrow 1] \\
&= \sum_{\mathsf{M}\in\mathcal{M}} P[\mathrm{SAMP1}^{\mathcal{A}} \Rightarrow 1 | \mathsf{M}^* = \mathsf{M}] \cdot p_m(\mathsf{M}) \\
&\leq \sum_{\mathsf{M}\in\mathcal{M}} P[\mathrm{SAMP0}^{\mathcal{A}} \Rightarrow 1 | \mathsf{M}^* = \mathsf{M}] \cdot (p_d(\mathsf{M}) + \frac{1}{2^{(h-\log_2 3)n}}) \qquad (2.16) \\
&\leq P[\mathrm{SAMP0}^{\mathcal{A}} \Rightarrow 1] + 3^n \cdot \frac{1}{2^{(h-\log_2 3)n}} \\
&= P[\mathrm{SAMP0}^{\mathcal{A}} \Rightarrow 1] + \frac{1}{2^{(h-2\log_2 3)n}}.
\end{aligned}$$

This theorem demonstrates that for given any sequence M, it is difficult for an adversary to tell whether it is sampled from p_m, or from p_d (with negligible advantage for appropriate h). This is critical because it essentially explains the power of honey encryption: the adversary is confused whether the decrypted sequence is the original one (from p_m) or a wrong one (from p_d). ■

$\mathrm{MR}^{\mathcal{B}}_{\mathsf{HE},p_m,p_k}$

$K^* \leftarrow_{p_k} \mathcal{K}$
$\mathsf{M}^* \leftarrow_{p_m} \mathcal{M}$
$C^* \leftarrow_{\$} \mathsf{HEnc}(K^*, \mathsf{M}^*)$
$\mathsf{M} \leftarrow_{\$} \mathcal{B}(C^*)$
return $\mathsf{M} = \mathsf{M}^*$

Figure 2.8: Game defining MR security. Given ciphertext C^* (encrypted from M^*), adversary $\mathcal{B}$ is allowed to guess the message by brute-force attack. $\mathcal{B}$ wins the game if his output message M is the same as the original message M^*.

The last step of the security analysis is the quantification of *message recovery (MR) security* for any adversary $\mathcal{B}$ against the encryption scheme HE.

Definition 2.2. *Let $\mathcal{B}$ be the adversary attempting to recover the correct sequence given the honey encryption of the sequence, as shown in Figure 2.8. The advantage of $\mathcal{B}$ against* HE *is*

$$\mathrm{Adv}^{\mathrm{mr}}_{\mathsf{HE},p_m,p_k}(\mathcal{B}) = P[\mathrm{MR}^{\mathcal{B}}_{\mathsf{HE},p_m,p_k} \Rightarrow \mathsf{true}].$$

We emphasize that p_k, the password distribution, is non-uniform. We assume the most probable password has a probability w. Using Lemma 2.1 and Theorem 2.1, we can establish the following theorem.

Theorem 2.2. *Consider* $\mathsf{HE}[\mathsf{DTE}, H]$ *(the detailed definition is available in [61]) with H (the hash function) modeled as a random oracle and* DTE *using an hn-bit representation. Let p_m be the sequence distribution with maximum sequence probability γ, and p_k be a key distribution with maximum weight w. Let $\alpha = \lceil 1/w \rceil$. Then for any adversary $\mathcal{B}$,*

$$\mathrm{Adv}^{\mathrm{mr}}_{\mathsf{HE},p_m,p_k}(\mathcal{B}) \leq w(1+\delta) + \frac{3^n + \alpha}{2^{(h-\log_2 3)n}}, \tag{2.17}$$

where $\delta = \frac{\overline{\alpha}^2}{2\overline{b}} + \frac{e\overline{\alpha}^4}{27\overline{b}^2}(1 - \frac{e\overline{\alpha}^2}{\overline{b}^2})^{-1}$ and $\overline{\alpha} = \lceil 3/w \rceil$ and $\overline{b} = \lfloor 2/\gamma \rfloor$.

Proof. The proof is similar to Corollary 1 in [61]. We omit the redundant details and specify the necessary modifications in the following.

p_m is a non-uniform sequence distribution and we assume $\gamma \leq 3 - \sqrt{5} \approx 0.76$, which is a requirement for Corollary 1 (in [61]). This assumption is reasonable considering the length of the sequence n (≥ 20000)[7]. To estimate γ, we can consider the sequence with all major alleles and pessimistically assume each major allele frequency is 0.995, large enough to give an upper bound for real datasets. Then, γ can be estimated by $0.995^{20000} \approx 2.89 \times 10^{-44} \ll 3 - \sqrt{5}$.

The term $\frac{3^n+\alpha}{2^{(h-\log_2 3)n}}$ is achieved by replacing $\mathrm{Adv}^{\mathrm{dte}}_{\mathsf{DTE},p_m}(\mathcal{A}) \leq \frac{1}{2^l}$ with our Theorem 1, and $|p_m(\mathsf{M}) - p_d(\mathsf{M})| < \frac{1}{2^l}$ with our Lemma 1 in the proof of Corollary 1 (in [61]). Essentially, $\frac{3^n+\alpha}{2^{(h-\log_2 3)n}}$ is the security loss due to DTE imperfectness that causes the difference between p_m and p_d.

[7]We need to focus only on one chromosome because there is no LD between chromosomes. The number 20000 is based on the observation of chromosome 22 (one of the shortest chromosomes) in a real dataset from the International HapMap Project.

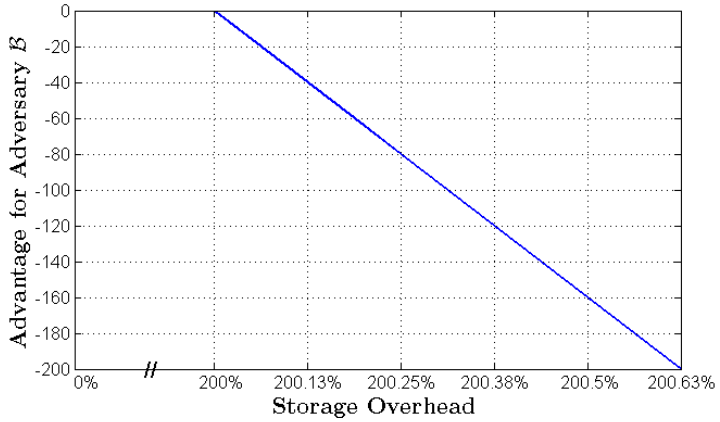

Figure 2.9: Adversary advantage versus storage overhead. Without encryption, the minimum storage for a sequence of n SNVs is $n \cdot \log_2 3$ bits. The x-axis is the expansion ratio between the storage with GenoGuard and the storage without encryption, namely, $\frac{hn}{n \cdot \log_2 3} = \frac{h}{\log_2 3}$. The y-axis is logarithm of the security loss term, $\log_2 \Delta_{\text{Adv}}$, that is part of the advantage of the message recovery adversary $\mathcal{B}$ (Equation (2.17)). With GenoGuard, to ensure a security loss smaller than 2^{-200}, we only need a storage expansion ratio that is slightly larger than 2.

By this theorem, even with a brute-force attack, an adversary's probability of succeeding in recovering a honey-encrypted sequence is close to the probability of the real password. Therefore, the attack does not help more than randomly guessing the password. ■

As mentioned in the proof, we denote $\Delta_{\text{Adv}} = \frac{3^n + \alpha}{2^{(h - \log_2 3)n}}$ as the security loss term. Consider a case where $n = 20000$, $h = 4$, and $\gamma = 2.89 \times 10^{-44}$. If p_k is a password distribution, then w can be estimated to be $1/100$ according to Bonneau's Yahoo! study [72], in which the most common password was selected by 1.08% of users. In this case, Δ_{Adv} is negligible ($\approx 2^{-16600}$), and $\delta \approx 0$, hence the upper bound on message recovery advantage is $w = 1/100$. If we consider an adversary who trivially decrypts the ciphertext with the most probable key and then outputs the resulting sequence, he can win the message recovery (MR) game with probability $1/100$. Hence, the bound is essentially tight. However, this case only happens if the patients choose weak passwords according to the previous password study.

To choose the storage overhead parameter h in practice, we consider how it affects the security loss term Δ_{Adv}. Since α is negligible compared to 3^n, we have $\Delta_{\text{Adv}} \approx \frac{1}{2^{(h - 2\log_2 3)n}}$. Taking the logarithm of Δ_{Adv}, we can observe that it has a linear relationship with h, as shown by Figure 2.9. For example, when $\frac{h}{\log_2 3} = 200.63\%$, we have $\Delta_{\text{Adv}} \approx 2^{-200}$. Hence, with a storage overhead slightly larger than two times (compared to the storage of a plaintext sequence), we achieve a negligible security loss.

Security under Brute-Force Attacks: To illustrate the security guarantee of GenoGuard, we conducted two experiments to compare GenoGuard with a simple (unauthenticated) PBE algorithm under brute-force attacks. For the simple PBE algorithm, we encoded the genome by assuming a uniform distribution in GenoGuard encoding, specifically by setting all edge weights in the tree to be equal (namely, $\frac{1}{3}$). Thus, its decryption

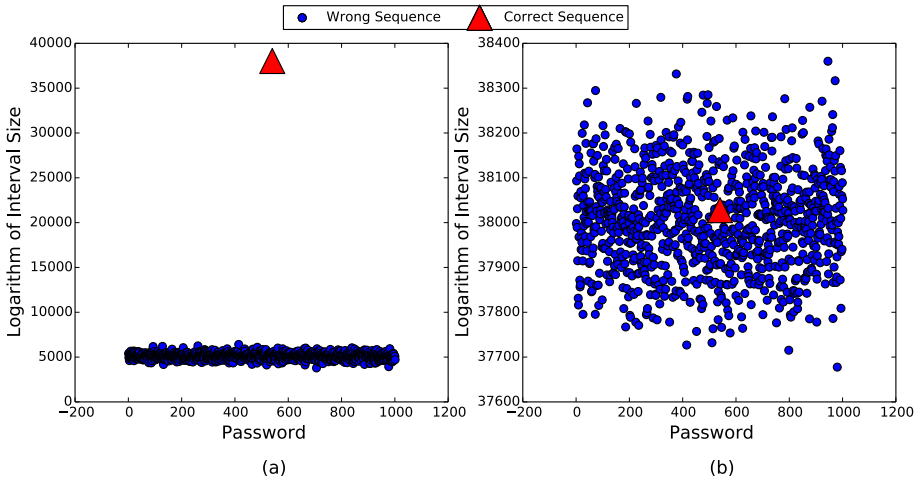

Figure 2.10: Experimental security evaluation. We encrypted a genome with a given password from a pool of 1000 passwords (for simplicity, we assume that the passwords are integers from 1 to 1000). Each point represents one decryption result using an integer from the password pool (the x-axis). The y-axis is the logarithm[8] of the interval size of the decrypted sequence when encoded with the recombination model. (a) With a conventional PBE scheme [64], all the wrong passwords have been ruled out except the correct one; (b) Obviously, with GenoGuard, no password can be excluded.

under any key yields a valid genome ("valid" does not necessarily mean "plausible", as we will show). We show here that for this PBE scheme a very simple classifier suffices for identifying the correctly decrypted genome with high probability. We encrypted a victim's chromosome 22 (see Section 2.6.1 for dataset description and implementation details) with a given password from a password pool of size 1000 (without loss of generality, we assume that the passwords are integers from 1 to 1000). We chose "539" as the correct password for both experiments; and we assumed that the adversary knows the correct password is a number from the password pool and that he performs a simple brute-force attack. In real life, brute-force attacks can be carried out if the adversary knows that the correct password has a limited number of characters (hence memorizable by users) or even a fixed length (e.g., six-digit PIN code). **In the first experiment**, we encrypted the victim's sequence directly with the PBE scheme in [64] (after encoding by assuming a uniform distribution). **In the second experiment**, we followed the same procedure except that we encrypted the victim's sequence by using the GenoGuard. Note that in our proposed DTE, the size of the interval of a leaf in the ternary tree is proportional to the probability of the corresponding sequence. In both experiments, to rule out wrong passwords, we computed the interval sizes of the decrypted sequences and observed the result. Figure 2.10 shows the result of the two experiments. We observe that if the sequence is protected by a direct application of the PBE scheme, the adversary can exclude most passwords in the attack because the corresponding decrypted sequences have much lower probabilities than that of the correct sequence. In this example, only the correct password is retained, as shown in Figure 2.10 (a). With GenoGuard, on the contrary, the correct sequence is buried among all the decrypted sequences, hence it is almost impossible to reject any wrong password.

2.5 Towards Phenotype-Compatible GenoGuard

An individual's physical traits (such as gender, ancestry and hair color) are highly correlated to his DNA sequence. Recently, researchers showed that it is even possible to model facial traits of an individual from his DNA [73]. Although such progress in human genetics is desirable for many applications (e.g., forensics), it can pose a threat to our proposed technique. In particular, such correlations could be used as side information by an adversary who tries to obtain the sequence of a specific victim (e.g., by trying various potential passwords). For instance, if the adversary knows that an encrypted sequence belongs to a victim of Asian ancestry, he might be able to eliminate a (wrong) password if the genetic sequence obtained using this password does not belong to an individual of Asian ancestry.

In genetics, gender and ancestry are the most well studied human genetic traits. These traits have deterministic genotype-phenotype associations, whereas other traits (such as hair color) have less certain (probabilistic) genotype-phenotype associations. In this section, we first show that the security of GenoGuard is not affected by traits with deterministic genotype-phenotype associations. Our main goal is to show that if an adversary knows a phenotype (physical trait) of a victim, he always retrieves a decrypted sequence that is consistent with the corresponding phenotype, even if he types a wrong password. Next, we quantify the privacy loss if an adversary has information about other traits (with probabilistic genotype-phenotype associations) of a victim via a privacy analysis.

2.5.1 Traits with Deterministic Genotype-Phenotype Associations

Gender: Gender is determined by sex chromosomes, namely, *X chromosome* and *Y chromosome*. Females have two copies of the X chromosome, whereas males have one X chromosome and one Y chromosome. Note however that X chromosome and Y chromosome have different lengths. Therefore, the adversary can immediately ascertain whether a ciphertext comes from an X chromosome or a Y chromosome because the latter is shorter than the former. As we mentioned in Section 2.3.3, (when implementing GenoGuard) the whole interval $[0, 2^{hn} - 1]$ is determined by the length n of the sequence. To deal with the gender problem, we use the length of X chromosome for both sex chromosomes. In other words, X chromosome and Y chromosome are encoded in the same interval $[0, 2^{hn} - 1]$, where n is the length of X chromosome.[9] In this way, the adversary cannot infer any information about the gender because the ciphertext is always of the same length, whether it belongs to a male sequence or a female sequence. Furthermore, if the adversary knows the gender of a victim, he will always get a consistent sequence (based on the gender) when he decodes the ciphertext by using the corresponding public knowledge of Y (or X) chromosome.

Ancestry: Research has shown that ancestry information can be accurately inferred from DNA sequences. For example, the sequence of an individual of Asian ancestry usually has different combinations of SNVs compared to an individual of European origin. In genetics, ancestry can be inferred with a number of methods, e.g., *principal compo-*

[8] Note that hn is close to 80000, hence the interval size is a huge integer and is better expressed as its logarithm with base 2.

[9] There is no LD between two different chromosomes, so each chromosome can be encrypted as an independent sequence.

nent analysis (PCA) followed by *k-means clustering* [74]. In this method, a training set is comprised of a number of individuals, each of which is genotyped on a predefined set of SNVs (the most informative SNVs). This training set is then fed into PCA in order to find several principal components. After the dataset is projected on these principal components, k-means clustering is applied to cluster the individuals into different ethnicities.

What we want to achieve in GenoGuard is ethnic plausibility: the principal components of the decrypted genome-wide genotyping data should be broadly similar to those from a real genome. Hence, we argue that the decoding operation with knowledge of recombination rates and haploid genotype dataset from a specific population always yields a sequence belonging to that population. To verify this, we conducted an experimental analysis depicted in the following.

We used Phase III[10] data from the HapMap dataset [70]. In this dataset, we chose 3 populations for our evaluation:

(i). ASW (African ancestry in Southwest USA), with 90 samples;

(ii). CEU (Utah residents with Northern and Western European ancestry from the CEPH collection), with 165 samples;

(iii). CHB (Han Chinese in Beijing, China), with 90 samples.

We selected 100 SNVs to infer ancestry according to [75]. First, we applied PCA on the above dataset and selected the first two principal components. The projection of the dataset on the two principal components can be seen in Figure 2.11(a). We encrypted a sequence from a specific population (e.g., ASW) by using GenoGuard. Then, for each of the three aforementioned populations, we decrypted the ciphertext with randomly guessed passwords 100 times, generating 100 random sequences for each case (in total, we generated 300 sequences). Finally, we projected these 300 sequences on the principal components and observed the result, as shown in Figure 2.11(b), (c), and (d). We conclude that decoding with public knowledge from a population always produces a sequence of that population, which proves that ancestry inferred from a sequence does not pose a threat to our proposed technique. We leave the case for people with mixed blood for the future work, but a reasonable assumption is that corresponding public knowledge could be available for mixed-blood people in the future.

2.5.2 Traits with Probabilistic Genotype-Phenotype Associations

In theory, the idea we introduce for ancestry also works for other traits: incorporate phenotype-related data during encoding. For the case of ancestry, such data is provided as population-specific haploid genotype dataset. However, such data is not easily available for many other traits (e.g., those with probabilistic genotype-phenotype associations) and genotype-phenotype associations is ongoing research. In the following, we quantify the privacy loss when the phenotype of a victim is not taken into account during encoding, but is exposed to the adversary as side information. For instance, the adversary could have access to a small number of phenotypical traits by observing a victim's photographs from online social networks.

[10]The third phase of the International HapMap project. This phase increases the number of DNA samples covered from 270 in phases I and II to 1,301 samples from a variety of human populations.

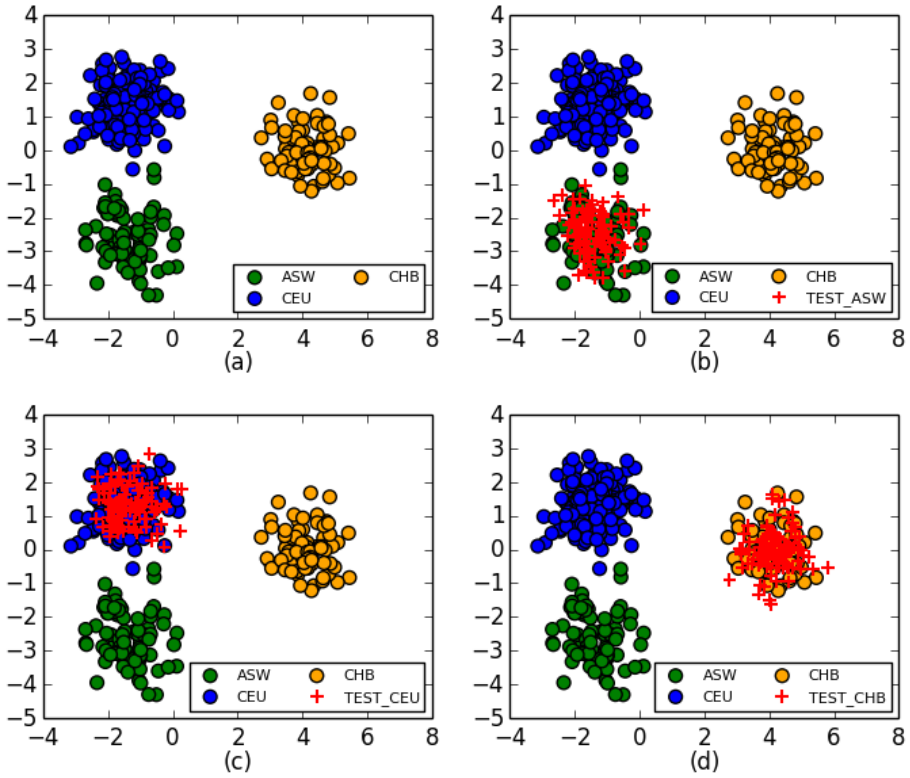

Figure 2.11: Evaluation of ancestry compatibility on GenoGuard. (a) Ancestry inference with PCA on three populations: ASW (lower left cluster), CEU (upper left cluster), and CHB (right cluster). The red crosses are sequences decrypted from an ASW person with randomly guessed passwords, but with public haploid genotype dataset from different populations: (b) ASW; (c) CEU; (d) CHB. We can see that, regardless of the population which the original sequence belongs to, the ancestry of the decrypted sequence only depends on population-specific haploid genotype dataset used for the decoding.

Consider a genetic trait that has a set of possible phenotypes $\{T_1, T_2, \cdots, T_u\}$. For example, the trait "hair color" can have phenotype set $\{\text{Red}, \text{Blond}, \text{Brown}, \text{Black}\}$. Let P_{T_i} denote the prior probability of a phenotype T_i. Each phenotype T_i is also associated with a vector of prediction probabilities $A_{T_i}^{T_j}$: given a sequence with phenotype T_i, $A_{T_i}^{T_j}$ is the probability that the best classification algorithm will associate the sequence with phenotype T_j. Then, a brute-force attack proceeds as follows. For each password, the adversary uses it to decrypt the ciphertext, inputs the result sequence to the classifier, and excludes the password if the phenotype does not match; otherwise he retains the password. We assume that the adversary trusts the classifier and makes a binary decision on whether he should retain the password.

Suppose there are totally N unique passwords at the beginning, and they are in descending order regarding their probabilities: $P_1 \geq P_2 \geq \cdots \geq P_N$. The order of a password is usually called its rank. Note that $\sum_{i=1}^{N} P_i = 1$. It has been shown that the distribution of real-life passwords obeys Zipf's law [76, 77]. In other words, for a

Hair Color (T^*)	Prior (P_{T^*})	$A_{T^*}^{Red}, A_{T^*}^{Blond}, A_{T^*}^{Brown}, A_{T^*}^{Black}$
Red	8.8%	$60.7\%, 28.6\%, 7.1\%, 3.6\%$
Blond	42.6%	$0.8\%, 93.9\%, 3.8\%, 1.5\%$
Brown	39.3%	$0.8\%, 56.7\%, 20\%, 22.5\%$
Black	9.3%	$0\%, 55.2\%, 3.4\%, 41.4\%$

Table 2.2: Summary of the results from the HIrisPlex system [78]. The second column, *prior*, is the fraction of samples that have the corresponding hair color. The third column is the vector of prediction accuracies (of the classification algorithm) for all four hair colors, given that a person has hair color T^*.

password dataset, the probability of password with rank i is

$$P_i = Wi^{-s}, \tag{2.18}$$

where W and s are constants depending on the dataset. This is actually the password distribution p_k. Suppose the victim's phenotype is T^*, which is known to the adversary. We assume that decryption under a given incorrect password yields phenotype T_i with probability P_{T_i}, and that such assignment is independent across passwords. Whether an incorrect password is retained then depends on the probability that the decrypted sequence is classified by the classifier as phenotype T^*. This event may be modeled as independent Bernoulli trials across passwords, each with retaining probability P_{ret} computed as

$$P_{ret} = \sum_{i=1}^{u} P_{T_i} \cdot A_{T_i}^{T^*}. \tag{2.19}$$

Note that for the correct password, the adversary retains it with probability $A_{T^*}^{T^*}$. From Theorem 2.2, we observe that the advantage of adversary $\mathcal{B}$ *without* side information is approximately equal to w, the maximum weight in the password distribution (equivalent to the above P_1). Let $\mathcal{B}'$ represent the adversary *with* side information T^*. $\mathcal{B}'$ first prunes passwords based on the classifier, and then executes the algorithm of adversary $\mathcal{B}$ in the MR game (Figure 2.8) on the resulting smaller password pool consisting of retained passwords. Let p'_k represent this new password distribution, with maximum weight w'. We can represent the password pruning procedure as a randomized function $f(p_k) \to p'_k$. Therefore, $\mathcal{B}'$ adheres to the procedure: i) $\mathcal{B}'$ uses f to compute p'_k; ii) $\mathcal{B}'$ gives p'_k to $\mathcal{B}$. Let $\mathrm{Adv}(\mathcal{B}')$ represent the advantage of adversary $\mathcal{B}'$. We have

$$\begin{aligned} \mathrm{Adv}(\mathcal{B}') &= A_{T^*}^{T^*} \cdot E_{p'_k \leftarrow f(p_k)}[\mathrm{Adv}^{\mathrm{mr}}_{\mathsf{HE},p_m,p'_k}(\mathcal{B})] \\ &\approx A_{T^*}^{T^*} \cdot E_{p'_k \leftarrow f(p_k)}[w'], \end{aligned} \tag{2.20}$$

where E is the expectation over the randomized password pruning process, and we approximate $\mathrm{Adv}^{\mathrm{mr}}_{\mathsf{HE},p_m,p'_k}(\mathcal{B})$ with the maximum weight w' in the password distribution p'_k. In the following, we quantify $\mathrm{Adv}(\mathcal{B}')$ empirically with real data.

For this purpose, we study a recent work about predicting hair color from DNA (the HIrisPlex system [78]). The study collects DNA samples and hair color information from 1551 European subjects and builds a model to predict the hair color. The results are shown in Table 2.2.

We use the Zipf's model in [77], where $N = 486118$, $W = 0.037871$ and $s = 0.905773$. For different hair colors known by adversary $\mathcal{B}'$, we perform the Bernoulli trials with

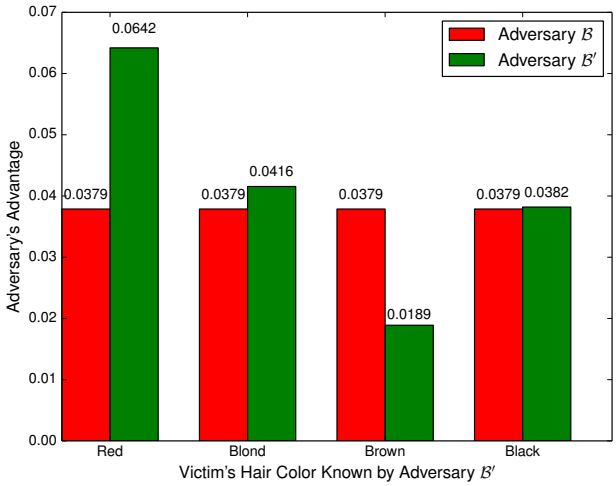

Figure 2.12: Evaluation of adversary's advantage with the side information of hair color. Adversary $\mathcal{B}$ has no side information, and his advantage is approximately $w = 0.0379$, the maximum weight in the original password distribution p_k. The advantage of adversary $\mathcal{B}'$ depends on the prediction accuracy A^{T*}_{T*} and the retaining probability P_{ret} for the victim's hair color T^*.

corresponding P_{ret} on the password pool, and estimate $\mathrm{Adv}(\mathcal{B}')$ in Equation (2.20). We repeat the whole experiment 1000 times for each hair color, and the average results are shown in Figure 2.12.

With the "Red" hair information, the adversary's advantage increases from 0.0379 to 0.0642, which is the worst among the four colors (for the victim). This is explained by the fact that "Red" hair has a very low prior probability that leads to a small P_{ret}, hence most wrong passwords are deleted. We observe that the empirical estimation of $E_{p'_k \leftarrow f(p_k)}[w']$ is consistently larger for a smaller P_{ret}. On the contrary, because "blond" hair has a high prior probability, a larger number of passwords are retained, hence smaller $E_{p'_k \leftarrow f(p_k)}[w']$ and smaller $\mathrm{Adv}(\mathcal{B}')$, compared to the case of "Red" hair. From Equation (2.20), the advantage is also positively correlated to the accuracy of the prediction algorithm. The low accuracies for "Brown" and "Black" hair (A^{Brown}_{Brown} and A^{Black}_{Black}) explain why the advantage of adversary $\mathcal{B}'$ barely increases, or even decreases[11], compared to adversary $\mathcal{B}$.

Even though side information is a common security concern in cryptography, we propose a general idea to avoid this problem for GenoGuard: incorporate the side information during the encoding phase. Nontrivial as this is, we will elaborate this idea in our future work, especially for traits with probabilistic genotype-phenotype associations.

2.6 Discussion

In this section, we discuss the performance, application scenarios, some extensions, and limitations of the proposed scheme.

[11] When the prediction is unreliable, it's better for the adversary to ignore the side information.

2.6.1 Performance

The time complexity of the encoding phase is $O(n)$ where n is the length of the sequence. Moreover, the storage overhead of the encrypted seeds is low as shown in Figure 2.9. Note that the ternary tree does not need to be stored. The encoding and decoding process are completely executed based on public knowledge; not on a pre-stored tree.

We implemented GenoGuard in Python. It includes mainly four steps: *encode*, *decode*, *PBE_encrypt*, *PBE_decrypt*. As before, we used the Phase 3 data in International HapMap Project, for the CEU population [70]. We set the storage overhead parameter $h = 4$. As for password-based encryption (decryption), we followed the standard PKCS #5 [64]. That is, using HMAC-SHA-1 as the underlying pseudorandom function, given a password P, we first applied a key derivation function

$$DK = KDF(P, S),$$

where DK is a 128-bit derived key and S is a 64-bit random salt. DK is used as the key for an AES block cipher that encrypts the seed in CBC mode. We ran the algorithm on a cluster of 22 nodes, each with 3.40GHz Intel Xeon CPU E31270 and 64-bit Linux Debian systems. In other words, the task of encrypting the whole genome was parallelized in 22 nodes that independently encrypt 22 chromosomes. We evaluated GenoGuard on 165 CEU samples, and the average performance is shown in Figure 2.13. Although encoding (decoding) is more costly than PBE, it is still acceptable considering the size of a full genome. Moreover, encoding different chromosomes was run in parallel, hence the running time depends only on the longest chromosome.

2.6.2 Application Scenarios

GenoGuard can be applied to various scenarios, including healthcare and recreational genomics, for the protection of genomic data. The general protocol in Figure 2.3 can work in a healthcare scenario without any major changes. In this scenario, a patient wants a medical unit (e.g., his doctor) to access his genome and perform medical tests. The medical unit can request for the encrypted seed on behalf of (and with consent from) the patient. Hence, there is a negotiation phase that provides the password to the medical unit. Such a phase can be completed automatically via the patient's smart card (or smart phone), or the patient can type his password himself. In this setup, the biobank can be a public centralized database that is semi-trusted. Such a centralized database would be convenient for the storage and retrieval of the genomes by several medical units.

For direct-to-customer (DTC) services, the protocol needs some adjustments. For instance, Counsyl[12] and 23andMe[13] provide their customers various DTC genetic tests. In such scenarios, the biobank is the private database of these service providers. Thus, such service providers have the obligation to protect customers' genomic data in case of a data breach. In order to perform various genetic tests, the service providers should be granted permission to decrypt the sequences on their side, which is a reasonable relaxation of the threat model because customers share their sequences with the service providers. Therefore, steps 8 and 9 in Figure 2.3 should be moved to the biobank. A

[12]https://www.counsyl.com/.
[13]https://www.23andme.com/.

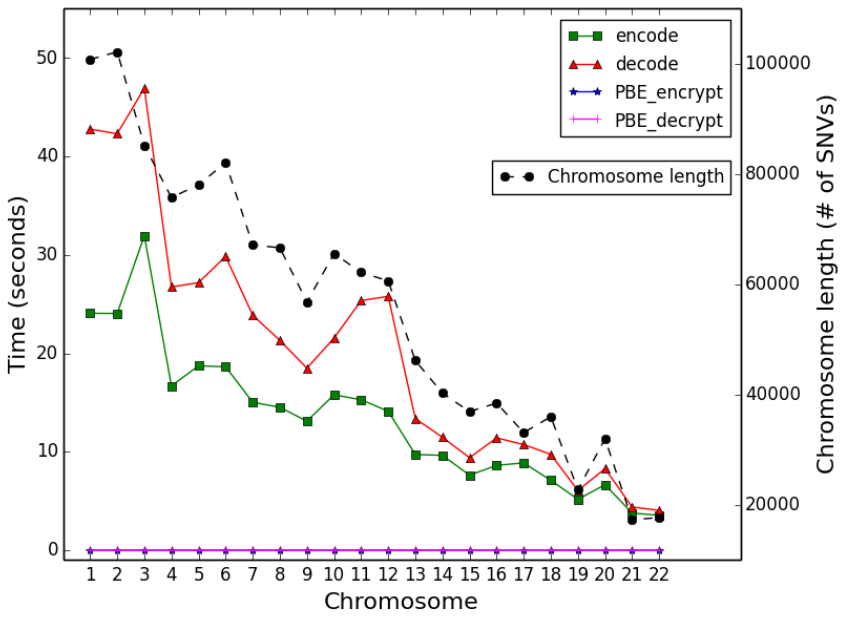

Figure 2.13: Performance of GenoGuard on 22 chromosomes, averaging over 165 CEU samples. The dashed line shows the length of each chromosome, whereas the solid lines show the running time of the four procedures: *encode*, *decode*, *PBE_encrypt*, *PBE_decrypt*. The number of SNVs roughly decreases from chromosome 1 to chromosome 22. We can see that the running time of password-based encryption (decryption) is negligible compared to encoding (decoding), whose running time increases almost linearly with the length of a chromosome).

user (customer) who requests a genetic test result logs into the biobank system, provides the password for password-based decryption and asks for a genetic test on his sequence. The plaintext sequence is deleted after the test.

2.6.3 Typos

Providing an incorrect password yields a fake but valid-looking sequence. This is a good security characteristic of honey encryption, but can be bad for usability if a legitimate patient or doctor does not realize she has made a mistake when typing the password. To solve this problem, we propose several solutions, as discussed below.

The first idea is to append to the plaintext some information that is unique and verifiable by the patient but meaningless for the adversary. We propose encoding such information (such as a 4-digit PIN chosen by the patient) as a string of bits similar to the seed. Such a PIN can be appended to the seed and encrypted together. In other words, the third encryption step in Figure 2.1 can be replaced with $C \leftarrow_{\$} \mathsf{encrypt}(K, S||\mathsf{PIN}, r)$. This option works well if the PIN is a uniformly random string; otherwise it will cause some security degradation because $S||\mathsf{PIN}$ is no longer uniform. Moreover, it requires the PIN to be kept secret and that the adversary cannot link it to a patient.

Another approach might be to leverage the distinction between recall memory and recognition memory [79]. The latter is shown to be more robust than the former. For instance, the system can provide a pool of N confirmation images, and the user can choose one before encryption. The confirmation images do not themselves have to be part of the ciphertext. The system can hash the genome sequence into $Z_N = \{0, 1, 2, \cdots, N-1\}$ to obtain a confirmation index, for security parameter N. The user might confirm correct decryption simply by indicating that a displayed image is familiar. A similar idea has been proposed in previous work where the authors apply it to anti-phishing techniques [80].

Another idea is based upon concealment of a biometric template among decoys. For instance, the user can provide his fingerprint template that is stored with some honey templates (e.g., synthetic fingerprint images [81], or other users' templates). These templates can also be indexed as what we propose for the confirmation images above. During retrieval, only the user can verify whether the decryption is correct or not using his own fingerprint.

2.7 Summary

The long-term sensitivity of genomic data gives rise to a need for especially strong protective mechanisms. Brute-force attacks on standard encryption schemes under strong passwords should not be considered infeasible in the long term, given the rapid evolution of computing technology and potential algorithmic advances. In the short term, the use of low-entropy keys, such as passwords, poses serious risks to password-based encryption of genomic data.

In this chapter, we describe GenoGuard, a cryptographic system that offers long-term protection for genomic data against even computationally unbounded adversaries. Decryption attempts against a GenoGuard ciphertext under an incorrect key yield a genome sequence that appears statistically plausible even to a sophisticated adversary. To achieve this guarantee, GenoGuard introduces a novel DTE scheme that efficiently encodes a genome sequence on a ternary tree with sensitivity to genetic recombination and mutation, thereby capturing the highly non-uniform probability distribution and special structure of genomic data. GenoGuard additionally provides security against adversaries with phenotypic side information (physical traits of victims). We provide a parallelized software implementation of GenoGuard and demonstrate its efficiency and scalability on a cluster of nodes. GenoGuard thus offers an appealing approach to the increasingly important challenge of protection of genomic data.

Chapter 3

Searching-enabled genomic data protection

In the previous chapter, we discussed the importance of securing genomic data storage, and presented a solution that is robust even under brute-force attacks against weak passwords. With either standard symmetric encryption or the presented GenoGuard, it is sufficient for scenarios where secure storage is the sole requirement and users always decrypt the whole data before usage (or decrypt data blocks if data is segmented into blocks which are encrypted separately with salted keys or with different nonces). In some use cases that we will describe later, efficient database searching is a necessary feature, but none of the aforementioned encryption options enable searching under semantic security. Moreover, in many genomic applications, searching is based on ranges, instead of keywords, which poses another design challenge for the reconciliation of security, functionality, and efficiency. In these applications, semantic security might be hard to achieve if one wants to satisfy all functionality and efficiency requirements and impose minimal changes to existing practices (which normally work with *NO* encryption at all). For example, to enable searching, one of the most secure and relatively efficient cryptography-based solution is *searchable encryption* [94]; nevertheless, it only works for keyword searching and leaks non-trivial information in searching results, and will bring excessive changes to current practices because big genomic data are stored in specific formats that are not necessarily compatible with searchable encryption schemes. In order to advocate a smooth transition from current insecure practices to the secure use of genomic data, in this chapter, we describe a privacy-preserving solution based on *order-preserving encryption* that provides relatively weaker notion of security than searchable encryption, but nonetheless is efficient and fully compatible with current practices.

While the generation of genome sequence data is no longer cost-prohibitive, the unprecedented rate of data production presents new challenges for data storage and management. For example, the 1000 Genomes Project Consortium generated more data in its first 6 months than the NCBI GenBank database had accumulated in its 21 years of existence [95]. Sequence data are being more routinely used for diagnostic purposes, which has raised concerns regarding security and privacy. Until recently, it was standard in clinical genetics to screen only one or two genes for mutations relevant to a spe-

cific disease, but high-throughput sequencing technologies have now made whole-genome or whole-exome sequence data commonplace. These comprehensive sequence data sets must then be securely stored and relevant variants made available to various stakeholders in the healthcare system. Preventing incidental leakage of personal data requires not only encryption but also defining access privileges and enabling selective retrieval. Although some encryption solutions (e.g., in cramtools [96]) have been proposed, they remain straight-forward applications of encryption standards and do not take into consideration the aforementioned threat model. Addressing these issues of security and privacy while minimizing storage costs will be essential for the large-scale application of personal genomics in research and clinical settings. Here, we describe a solution that minimizes information leakage, stores the sequence data in a lossless compressed format, and optimizes the performance of downstream analysis (e.g., variant calling).

Since 2007, when the first high-throughput sequencing technology was released to the market, the growth rate of genomic data has outpaced Moore's law, more than doubling each year [97]. Big data researchers estimate the current worldwide sequencing capacity to exceed 35 petabases per year [98]. For every 3 billion bases of human genome sequence, 30-fold more data (about 100 gigabases) must be collected to ensure sufficient coverage at each nucleotide. More than 100 petabytes of storage are already used by the worlds largest 20 biological research institutions; this corresponds to more than $1 million USD in storage maintenance costs if we consider Amazon cloud storage pricing [99]. This number continues to grow, and it is estimated that 2 - 40 exabytes of storage capacity will be needed by 2025 to store hundreds of thousands of human genomes. To face this challenge, more efficient approaches to genomic data storage are needed.

Current approaches for genomic data storage use different methods for compression [100]. Before high-throughput technologies were introduced, algorithms were designed for compressing genomic sequences of relatively small size (e.g., tens of megabases). These algorithms, such as BioCompress [101], GenCompress [102], and DNACompress [103], exploit the redundancy within DNA sequences and compress the data by identifying highly repetitive subsequences. The latest sequencing technologies pose new challenges for the compression of genomic data in terms of data size and structure. Due to the high similarity of DNA sequences among individuals, it is inefficient to store and transfer a newly assembled genomic sequence in its entirety, because $> 99\%$ of the data for two assembled human genomes are the same. This has led to the approach of storing only differences from a reference sequence (known as reference-based compression), such as the DNAzip algorithm [104]. Apart from entire assembled sequences, individuals sequence data are typically organized as millions of short reads of 100 to 400 bases, as produced by state-of-the-art sequencing technologies. Each genomic position is usually covered by multiple short reads. General-purpose compression algorithms, such as gzip [105], are applicable to these data sets. For example, the BAM format [106], which remains the de facto standard for storing aligned short reads, is already highly compressed by applying gzip compression to the data blocks.

Various advanced compression algorithms have been proposed for high-throughput DNA sequence data (Quip [107], Samcomp [108], HUGO [109], etc.). Among larger data sets (e.g., The 1000 Genomes Project Consortium), there is an observable trend toward using the highly compressed format CRAM [110], a reference-based compression algorithm for aligned data. Most advanced algorithms also use variable-length encoding (VLC) techniques, such as Huffman encoding and Golomb encoding, to compress

the metadata (read name, position, mapping quality, etc.). Recently, researchers developed a cloud-computing framework called ADAM [111], which uses various engineering techniques (e.g., dictionary coding, gzip compression, distributed processing) to reduce storage costs by 25% compared with BAM. This scheme achieves significant (2×-10×) acceleration in genomic data access patterns. A group of MIT researchers released a lossy compression algorithm called Quartz [112], which can discard 95% of quality scores without compromising the accuracy of downstream analysis. Their method works on the FASTQ file stage and can be plugged in as a preprocessing step for the aforementioned methods (and our solution) to achieve a higher compression ratio. Our solution is a lossless compression method that enables the user to completely reconstruct the original data (BAM files). To our knowledge, there is no existing compression solution that provides strong security and privacy control, the issue we address in this chapter. There is thus a need to integrate encryption methods into compression solutions for genomic data that are secure and privacy-preserving [100]. The closest example of such a solution [115] provides a privacy-preserving solution for storing BAM files, but it does not provide an efficient method for compression.

For clinical or research purposes, the most valuable information from human genomic data is the set of genetic variants that are identified across the genome. Typically, sequence data are taken as input for a pipeline and retrieved for downstream analyses, for example, variant calling. Given this usage scenario, it is crucial to have a storage format that is amenable to efficient downstream analyses. For example, it is common practice to aggregate information on a position from all short reads that cover the position (pile-up); hence, it is desirable that the storage format organizes information in this manner. In addition, it is challenging to enable selective retrieval of encrypted data because global encryption solutions obfuscate index information. As we demonstrate in this chapter, our SECRAM solution is convenient not only for retrieval but also for protection in that it enables the retrieval of specific information about the genome without compromising the rest of it.

In this chapter, we present our genomic data storage solution to address the challenges of compression, security, and retrieval. SECRAM is a novel aligned data storage format that is (1) organized in position-based storage that enables random queries anywhere in the genome, (2) highly compressed through a combination of reference-based and general data compression techniques, and (3) encrypted with standard secure cryptographic techniques and a fine-grained privacy control mechanism. By applying our solution, sequence data are securely protected in storage and can be efficiently retrieved for downstream analysis (e.g., variant calling) without any access to unauthorized information. Below, we compare SECRAM with two state-of-the-art storage solutions, BAM and CRAM, which are two of the most widespread formats for aligned sequence data. Our major goal is to bridge the gap between compression and protection of genomic data.

3.1 Background

The DNA sequence data produced by DNA sequencing consists of millions of short reads, each typically including between 100 and 400 nucleotides (A,C,G,T), depending on the type of sequencer. These reads are randomly sampled from a human genome. Each read is then treated and positioned (aligned) to its genetic location to produce a so-called *SAM* file (*Sequence Alignment Map*). BAM file (*Binary Alignment Map*) is simply the

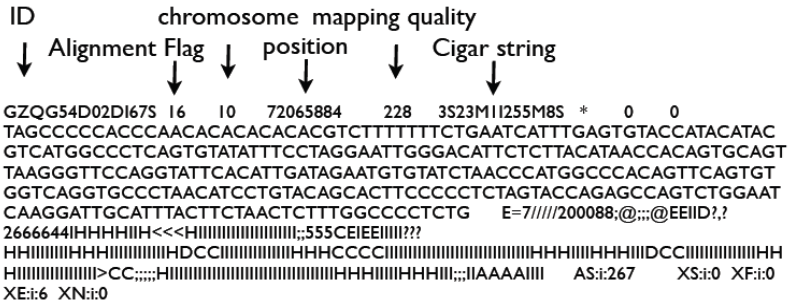

Figure 3.1: Format of a short read in SAM/BAM.

binary version of SAM, including some compression techniques. There are hundreds of millions of short reads in the SAM file of one patient. In Figure 3.1, we illustrate the format of a short read in a SAM file.

We focus the introduction on positions, cigar string, and nucleotide content $\{A, T, C, G\}^*$. A short read's position denotes the position of the first aligned nucleotide in its content, with respect to the reference genome. The cigar string (CS) of a short read expresses the variations in the content of the short read, with respect to the reference genome. For example, '15M' means the next 15 nucleotides match (or mismatch with substitution) those on the reference genome, while '2I' means the next 2 nucleotides are inserted with respect to the reference genome. For more explanation about the format, the readers are referred to the SAM specification [119].

3.2 System model

SECRAM is essentially a storage format that could be used locally like BAM. Users can employ the API to efficiently extract any region of data from the file without decrypting and decompressing the whole file.

However, in order to provide strong security and privacy guarantees in a cloud computing scenario where no single party (e.g., the storage server, the client) is trusted, we envision the following system for efficient information retrieval by using SECRAM. We adopt the system architecture proposed by Ayday et al. [115]. We provide below a brief summary of this system.

(i). *Patient (P)* goes to a *Certified institution (CI)* for genome sequencing. *CI* sends the sequence data to a centralized database *Biobank*, and the secret keys to *Masking and Key Manager (MK)*.

(ii). *Medical Unit (MU)* sends a query to *Biobank* retrieve data of a patient in the position range $[P_1, P_2]$.

(iii). *Biobank* communicates with *MK* to acquire **encrypted** and **masked** decryption key stream for the range $[P_1, P_2]$. Finally, *Biobank* sends back the ciphertext and sanitized decryption key stream to *MU*.

The third step will be explained in the next two sections with detailed encryption and masking techniques. For the sake of brevity, we mention the security and privacy goals we intend to achieve in the system:

- The original content of a short read cannot be accessed without explicit permission from the owner;
- Users (e.g., medical units) can access only restricted regions of data that are authorized by the owner; each user is allowed to access only specific region(s), e.g., a medical unit that specializes in female breast and ovarian cancers might be allowed to access only BRCA1 and BRCA2 genes;
- In order to prevent the inference attack on the purpose of user access, the storage center (i.e., the Biobank) should not learn about the region that a user accesses.

3.3 SECRAM Format

The SECRAM format conversion framework is depicted in Figure 3.2, including transposition from read-based storage (BAM) to position-based storage, compression, encryption, and retrieval (e.g., for variant calling). Following sequence alignment to a reference genome, the data are transposed, compressed, and encrypted. All these steps are one-time operations for each individual file. Afterwards, the SECRAM format can be queried routinely for data retrieval. More details of SECRAM are specified in the following sub sections.

3.3.1 Transposition

Aligned genomic sequence data are typically stored by sequencing reads (e.g., BAM, CRAM formats) (Figure 3.2-A). The first step of our solution is to organize the data by position instead of by read (Figure 3.2-B). This is crucial because (1) it facilitates multiple downstream analysis procedures, notably the variant calling pipeline, and (2) it allows us to seamlessly combine compression and encryption (fine-grained privacy control). The conversion between the two formats is equivalent to a matrix transposition, when we consider the index of the reads as one dimension and the position of the reference genome as the other dimension. If necessary, our format can be inversely transposed to BAM without losing information; this functionality makes our format compatible with several other applications designed for a read-based format, such as visually displaying reads that overlap with a specific genomic region.

Figure 3.3 describes the detailed structure of SECRAM in the manner of regular expressions. We denote the important notations as follows:

- '^' is the start marker of a read
- '$' is the end marker of a read
- '.*' denotes the base information at this position.
- '[Read Info]' contains information about this read, such as read name (or id), strand, mapping quality, etc.

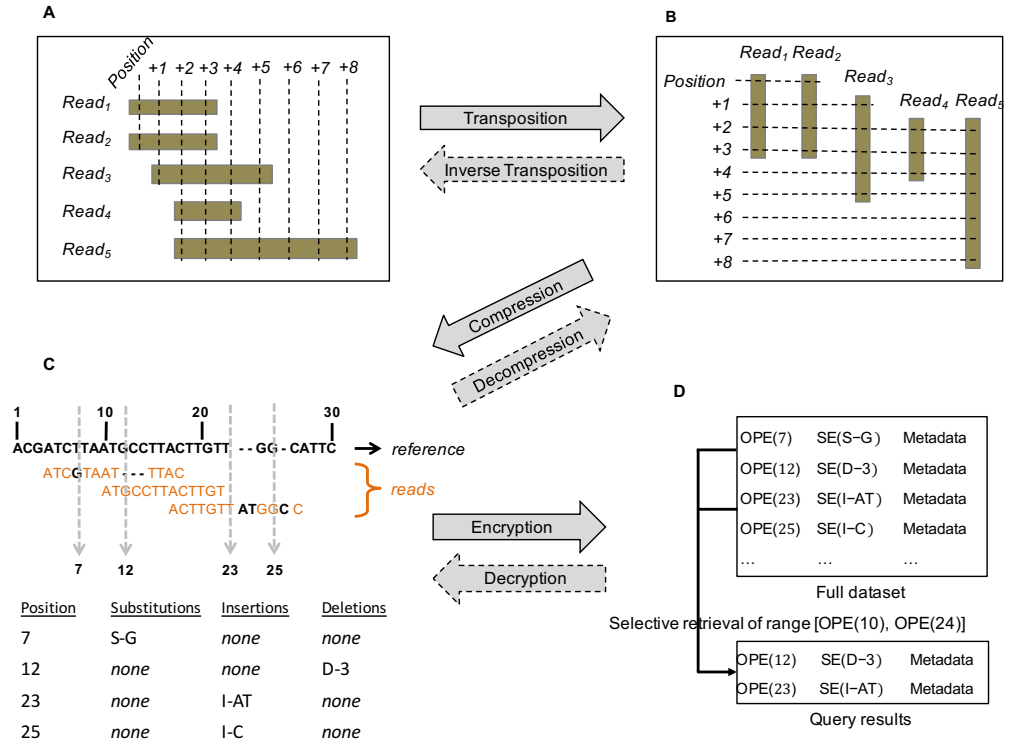

Figure 3.2: SECRAM framework for compressed, encrypted storage of genomic data. (A) The sequencing read-based format used in BAM is transposed into a genome position-based format (B). The read-based format can be reconstructed from the position-based format via reverse transposition. (C) The position-based storage is compressed and decompressed using a reference-based compression technique. In the table, "S-G" stands for substitution with base "G", "D-3" stands for deletion of three bases, and "I-AT" stands for insertion of two bases "AT". (D) The compressed position-based storage is encrypted to generate the final SECRAM format using order-preserving encryption (OPE) and traditional symmetric encryption (SE) scheme. "OPE (POSITION)" represents the OPE ciphertext of POSITION, and "SE(VARIANT)" represents the SE ciphertext of VARIANT. Metadata are not encrypted (e.g., quality scores, mapping quality, read name). The compressed format is recovered from the encrypted format by running the respective decryption algorithms. Our encryption enables efficient selective retrieval. For instance, if a user wants to retrieve data in the range [10, 24], the database executes a normal search between OPE(10) and OPE(24) based on the order-preserving property of OPE, and in the shown example, two positions, OPE(12) and OPE (23), are returned.

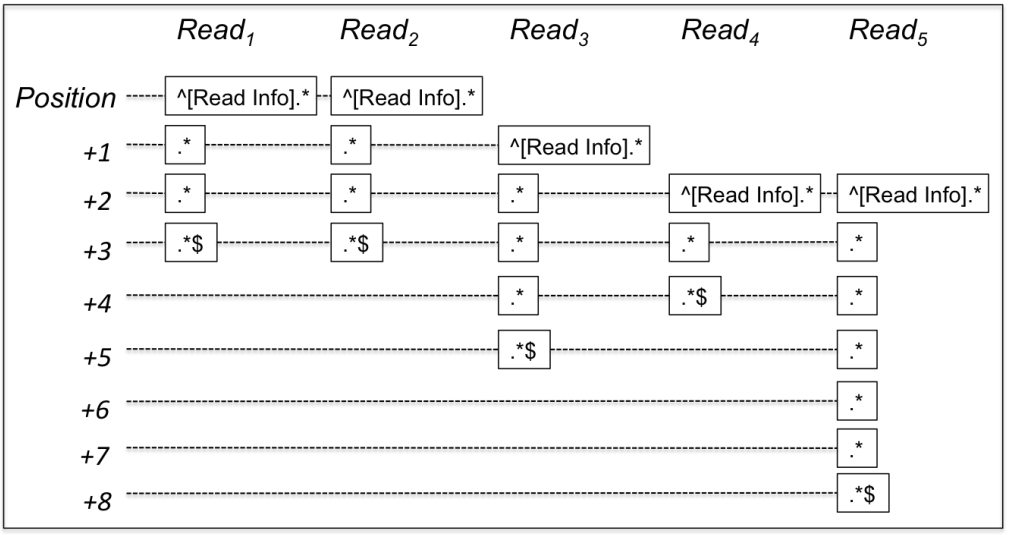

Figure 3.3: The basic structure of SECRAM. '^' is the start marker of a read. '$' is the end marker of a read. '.*' denotes the base information at this position. '[Read Info]' contains information regarding this read, such as read name (or id), strand, mapping quality.

3.3.2 Compression

Our algorithm takes advantage of several efficient compression methods used in CRAM:

- Reference-based compression.As shown in Figure 3.2-C,we use reference-based compression and store only the differences at each position relative to that of a chosen reference sequence.
- VLC. This technique is used to further compress the differences found in reference-based compression along with read metadata, such as the mapping quality. Depending on the information content, the most efficient encoding technique is selected, for example, Huffman, Golomb, and Beta encodings [96].
- Block compression. After the previous compression phases, positions are grouped into blocks and compressed with gzip. Blocks are gzipped separately, so that they can be decompressed independently for fast random access. This phase is important for the compression of information such as read name, quality score, and other auxiliary text.

Reference-based compression

The basic structure in Figure 3.3 does not take the redundancy of short reads into account, as a large portion of most reads are likely to match the reference. Reference-based compression is applied in most compression methods. To explain how we apply the idea in SECRAM, we make use of a new concept called position cigar string, PosCigar for short. For a given position, if we imagine that the reads that cover the position are ordered by their starting positions, then each read is attached with a unique order. Note

that one read could have different orders for different positions because the corresponding lists of covered reads in those positions vary. For example, in Figure 3.3, on Position +3, "$Read_3$" has order 3 because it is the third covering read, but on Position +4, it has order 1 because "$Read_1$" and "$Read_2$" end in the last position and "$Read_3$" now becomes the first covering read. Given a position, a PosCigar is composed of mainly three possibilities:

(i). Order$||'S'||[A|T|C|G]$: the read (specified by Order) has a substitution with the specified letter compared to the reference.

(ii). Order$||'I'||i||\{A,T,C,G\}^i$: the read has an insertion of i letters that are listed.

(iii). Order$||'D'$: the read has a deletion.

For example, a PosCigar that looks like "9I4ATTG...23SA...57D", means:

- 9I4ATTG: an insertion of 4 letters "ATTG" in the 9th read
- 23SA: a substitution with letter 'A' in the 23rd read
- 57D: a deletion in the 57th read

More operators (e.g., soft clipping, hard clipping, skipping region [106]) are also encoded in SECRAM, and we omit the redundant details here because each of them is handled similarly as one of the three aforementioned operators. As one possible design option, each position row can be encoded as:

Row size||Position||List of read headers||quality scores||PosCigar

In the following, we explain the terminology used in this option.

- **Row size**: Length (measured in bytes) of a position row;
- **List of read headers**: List of information of the reads that start at this position. It is decomposed as "(Order||Read Info)$*$" , with '*' meaning an arbitrary number of such headers. The read information would also include read length so that we do not have to store the end marker as in Figure 3.3.
- **Quality scores**: Quality scores for the bases of this position.
- **PosCigar**: The variants information.

3.3.3 Encryption

SECRAM protects genomic data using secure symmetric encryption (SE) techniques. To provide fine-grained privacy control and avoid information leakage in data retrieval, our solution encrypts the variant information for each position independently. We encrypt positions with order-preserving encryption (OPE) [120] to enable efficient retrieval of encrypted data without decryption. This is a critical feature to prevent insider attacks on servers that store the encrypted data. We encrypt other sensitive information (e.g., short read differences relative to the reference) using conventional SE (e.g., AES). Figure 3.4 summarizes the format encryption design. Notably, encryption adds randomness to the data, which inevitably affects the compression ratio. We explain the basics of these two encryption schemes below.

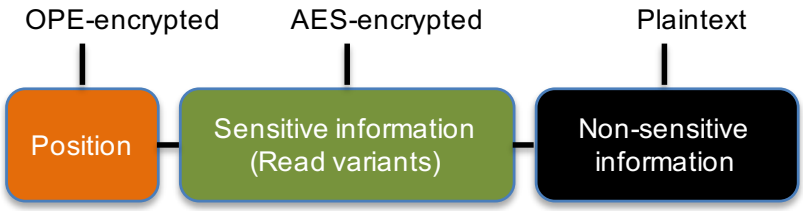

Figure 3.4: Protection methods for different types of information of a position row.

Symmetric encryption

In SECRAM, we make use of the stream cipher mode [121] of SE, which is done in two steps: (1) By using the encryption key, the data sender generates a random bit stream of identical length to the data; and (2) the encryption is performed via bitwise XOR of the stream with the data. The decryption is performed similarly: By using the decryption key, the ciphertext receiver first generates the bit stream as the sender does, and then decrypts the ciphertext via bitwise XOR of the ciphertext with the stream. The stream cipher mode enables the server to tailor the decryption key stream to only those positions matching the data retrieval request, hence sensitive information of other positions is also protected from clients who send the data request. With our solution, the privacy control is precise at the position level, in contrast to some existing solutions that provide coarse-grained encryption on the data, which leaks non-negligible information in each data retrieval. For example, if data region A, B, C from user X are in the results, but user X only authorizes to reveal region A and C, then region B of the decryption key stream will be replaced with random bits. The process is explained in detail in Section 3.4.

Order-preserving encryption

In a traditional SE scheme, to an observer who does not have the encryption key, the ciphertext is indistinguishable from a random bit stream. Therefore, the ciphertext does not reveal any useful information about the underlying data. OPE differs from traditional SE, however, in that the OPE ciphertext reveals both the order and equality information of the underlying data, namely, if $m_1 \leq m_2$, then $\mathsf{OPE}(m_1) \leq \mathsf{OPE}(m_2)$.

To understand the encryption design of SECRAM, it is important to understand the challenges of securing existing formats (e.g., CRAM). Consider a straightforward, blockwise SE solution for CRAM-formatted data (Figure 3.5-A). The problem is that the solution leaks information on other positions outside of the query range during retrieval and decryption. As a block in CRAM usually contains multiple reads, it is usually the case in practice that the retrieved block reveals parts of reads outside of the query position range [115]. Even if encryption is performed read-wise, the possibility remains that a single read can leak information on positions outside of the query range.

Figure 3.5 shows how encryption is applied in SECRAM, with each block encrypted based on positions. We encrypt the position information using OPE; the compressed sensitive content at the respective positions, using SE. Therefore, during retrieval, our scheme only outputs the sequence and metadata in the query range without the risk of revealing any information about undesired (or unauthorized) positions.

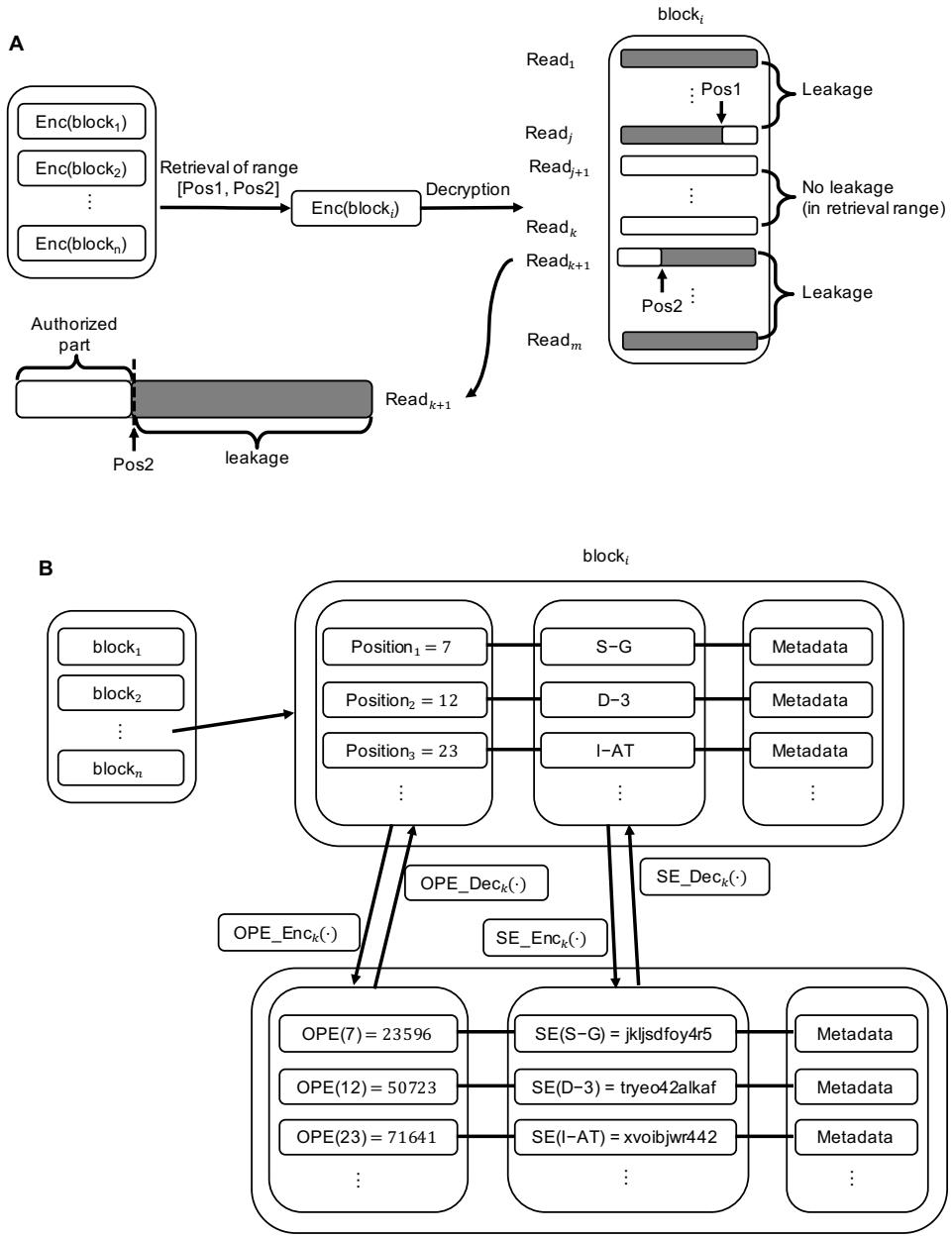

Figure 3.5: Solutions for encrypting genomic data. (A) A standard SE solution on CRAM-compressed data. It leads to potential data leakage when querying specific genomic regions. The encryption is performed over each individual data block. As a block in CRAM usually contains multiple sequencing reads, it is usually the case that the retrieved block will reveal reads or positions that are not in the query position range. (B) SECRAM encryption. Our solution encrypts each block in SECRAM format based on positions. Position information is encrypted with OPE; the compressed content at each position, with SE. This ensures that only information corresponding to the query position range is retrieved and decrypted. "OPE(POSITION) represents the OPE ciphertext of POSITION, and "SE(PosCigar)" represents the SE ciphertext of PosCigar. Metadata are not encrypted (e.g., quality scores, mapping quality, read name). Note that OPE preserves the order of the positions, namely, $23596 < 50723 < 71641$ because $7 < 12 < 23$, but SE encrypts the original message to a random string, for example, from "S-G" to "jkljsdfoy4r5".

Metadata

The SECRAM format contains all necessary information to enable reconstruction of the original BAM files. This is achieved with the metadata field at each position. This field is not encrypted and contains two categories of information: (1) quality scores and (2) information about reads that begin at that position. The first category is a numerical score (the phred-scaled base error probability: $-10\log^{10} Pr\{\text{base is wrong}\}$), while the second has a slightly more complicated structure. For each read that begins at position P, the metadata for P will store the following information: read name, read length, mapping quality, and pair information.

3.3.4 Indexing

The indexing of aligned genomic data is a map from positions to file offsets such that, given queried positions, a data reader can randomly access the file. Indexing is easy to implement in SECRAM. Our index file contains a list of tuples of position and file offset, where position is the genomic position of the first position row in a gzip block, and file offset is the byte offset of a gzip block in the compressed file. Hence, when we retrieve data for a position, a binary search first locates the gzip block containing the position, and then a linear search locates the position in the block.

3.4 System Implementation

In this section, we provide details about the encryption schemes, key management, and protocols for secure and fast retrieval.

3.4.1 Encryption

All fields, except "Position" and "PosCigar", are left in clear, because they do not contain any highly sensitive information. The encryption scheme can be described as follows:

- The patient (data owner) is assigned with a master key K_m that is used to derive various encryption keys.
- The position field is encrypted with order-preserving encryption (OPE) [120] for the efficiency of a random access based on position comparison (details in Section 3.3.3). A key, K_{ope}, is derived from K_m and used for OPE.
- The PosCigar field is encrypted with a stream cipher (SC). A key, K_{sc}, is derived from K_m and used for the SC. For the i-th GZIP block in the file, the PosCigar fields of all the included position rows are concatenated to generate a single data block, D_i, which is encrypted with K_{sc} and a 64-bit random salt R_i. The random salts are stored in the index file.

The above keys are stored at the MK.

3.4.2 Basic Retrieval

The retrieval process consists of the following steps:

(i). MU and MK setup a one-time symmetric session key K_{ss};

(ii). MU encrypts the range $[P_1, P_2]$ with AES under the key K_{ss} , to generate the ciphertext $[\mathsf{AES}(K_{ss}, P_1), \mathsf{AES}(K_{ss}, P_2)]$ that is sent to MK (or first sent to Biobank which forwards it to MK);

(iii). MK decrypts $[\mathsf{AES}(K_{ss}, P_1), \mathsf{AES}(K_{ss}, P_2)]$ and re-encrypts it with OPE under the key K_{ope}, generating ciphertext $[\mathsf{OPE}(K_{ope}, P_1), \mathsf{OPE}(K_{ope}, P_2)]$ that is sent to the Biobank;

(iv). The Biobank retrieves the k GZIP blocks $(B_i, B_{i+1}, \cdots, B_{i+k-1})$ that overlap with the range $[\mathsf{OPE}(K_{ope}, P_1), \mathsf{OPE}(K_{ope}, P_2)]$, with the help of the index file. The Biobank decompresses the block B_i, and computes an offset value of the position row $\mathsf{OPE}(K_{ope}, P_1)$, denoted by OFF_{P_1}: the byte offset of its PosCigar field in the data block D_i (D_i was defined in Section 3.4.1). OFF_{P_2} is computed similarly. The Biobank sends OFF_{P_1}, OFF_{P_2}, and random salts $(R_i, R_{i+1}, \cdots, R_{i+k-1})$ to MK;

(v). With key K_{sc}, MK derives the corresponding decryption keys $(DK_i, DK_{i+1}, \cdots, DK_{i+k-1})$ for the k blocks, masking those rows out of the range $[P_1, P_2]$. MK encrypts these keys, thus generating $(\mathsf{AES}(K_{ss}, DK_i), \mathsf{AES}(K_{ss}, DK_{i+1}), \cdots, \mathsf{AES}(K_{ss}, DK_{i+k-1}))$, and it sends these encrypted keys to the Biobank;

(vi). The Biobank sends the k GZIP blocks $(B_i, B_{i+1}, \cdots, B_{i+k-1})$ and the encrypted keys $(\mathsf{AES}(K_{ss}, DK_i), \mathsf{AES}(K_{ss}, DK_{i+1}), \cdots, \mathsf{AES}(K_{ss}, DK_{i+k-1}))$ to MU;

(vii). MU decompresses the blocks and decrypts the content.

3.4.3 Context-Aware Retrieval

In the basic retrieval scheme, we do not address the problem that the returned data might contain incomplete reads and thus cannot perform queries of read information, such as mapping quality, strand, and a relative position inside a read. Because the metadata (read name, mapping quality, read flags, etc.) are stored only at the starting position of a read, Biobank needs to retrieve the complete context of a position row. One solution is to return the "Read Info" of the position row.

Consider the query range $[P_1, P_2]$. If the position row P_1 contains any read that is not complete in block B_i (specifically, it does not start at this block), the storage server traces back to the previous block(s) and crops the corresponding metadata fields for the incomplete reads of row P_1. As a block normally contains thousands of positions, the server usually needs to look back to at most one previous block, as long as the reads are not excessively long (e.g., in our experiments, we use a block size of 50,000 positions). The server then returns these metadata fields along with the complete sequence data (only) within the query range.

3.4.4 Efficient Query Processing

We discuss how to efficiently answer the queries that are interesting to MU. Before going into details, we state the following assumption:

- The genomic positions are queried sequentially. In other words, if the position rows in a genomic range $[P_1, P_2]$ are returned, these rows are queried from P_1 to P_2 one

by one. This assumption does not affect the correctness of our system, rather it affects the efficiency.

The queries related to one position are more efficient than those related to a complete read. Examples of queries of one position include the coverage of a position, the variants (substitution, insertion and deletion) and quality scores. The complexity of answering these queries is $O(C)$, where C is the coverage of this position. The overall complexity of iterating all positions in the range $[P_1, P_2]$ is $O((P_2 - P_1)C)$.

For queries related to a complete read, we need to access the "Read Info" field. But this field does not exist in each row. The main idea is that we run the reconstruction algorithm for a position row whenever these queries are needed. Then the complexity of answering these queries for all rows in the range $[P_1, P_2]$ is $O((P_2 - P_1)CL)$, where L is length of a read and C is the coverage. But with the above assumption, we can carry out the task much more efficiently. In the context-aware retrieval, the Biobank needs to run the reconstruction algorithm for the position P_1. In the data returned to MU, "Read Info" fields already exist for all the reads of position P_1. To answer queries from position P_1 to P_2, MU maintains a cache that contains the "Read Info" fields of all reads in the current position. The cache is initialized to the "Read Info" fields of position P_1, and is updated as MU move to the next position. The complexity of this method is $O(CL + (P_2 - P_1)C)$, if the one-time reconstruction in the Biobank is taken into account.

3.4.5 Software Details

For processing BAM files and applying VLC and block compression techniques, SECRAM uses the open-source Java library, HTSJDK [122], designed for high-throughput sequencing data. For encryption, SECRAM builds its security solution based on the open-source Java library, Bouncy Castle [123]. Depending on the category of information, SECRAM chooses appropriate compression methods to achieve high compression ratio (Table 3.1). SECRAM is open source at https://github.com/acs6610987/secram.

3.5 Evaluation and Analysis

3.5.1 Storage Analysis

In Figure 3.6, we show the compression performance of SECRAM compared with BAM and CRAM for both paired and unpaired simulated data sets of different coverages[1] and error rates (substitution, insertion, and deletion). Both the SECRAM and CRAM formats preserve all information from BAM files, including quality scores, read names, and read-pair information. The number of bits per base is calculated by dividing the size of the file by the total number of bases (roughly equal to reference length multiplied by the coverage). Unsurprisingly, for all three formats, the average per-base storage cost decreases as the coverage increases. In contrast to CRAM, SECRAM does not compress efficiently when the coverage is very low (e.g., 1×) because of the storage overhead of encryption. Yet as coverage increases (e.g., 10×), the benefit of compression becomes more pronounced, mitigating the storage costs of encryption. In the best case, for high-coverage unpaired data with a 0.01% error rate, SECRAM and CRAM have almost

[1] Coverage is the number of reads that include a given nucleotide in the data file.

Information type	**Compression/encryption method**
Position	Step 1: Order-preserving encryption Step 2: Gzip block compression
Read bases	Step 1: Reference-based compression and VLC compression Step 2: AES encryption in CRT mode
Quality scores	Gzip block compression (optional lossy compression)
Read name	Gzip block compression
Mapping quality	VLC compression
Template length	Gzip block compression
Flag (e.g., strand)	VLC compression
Auxiliary text (e.g., tags in BAM)	Gzip block compression

Table 3.1: Compression and encryption methods for different types of information in SECRAM. Only positions and read bases are encrypted, with OPE and SE, respectively. Note that positions are encrypted with OPE before block compression in order to enable indexing. Block compression is chosen for the information that has a wide variety of values, or for text information, or merely as a default way to save storage. AES encryption in CTR mode stands for advanced encryption standard in counter mode, which is one way of applying a block cipher in a stream-cipher mode (essentially equivalent to a stream cipher).

identical compression ratios, both using nearly 4 bits/base. We also observed that, in general, compression ratios are lower for paired data with higher error rates because there is more information (e.g., metadata and variants) in the data. Compared with BAM, SECRAM saves 18% of storage on average when the coverage is higher than 10×; this is slightly lower than CRAM, which saves 34% of storage but has no privacy and security features. Figure 3.6-B shows the average storage costs on several randomly selected real data files from the 1000 Genomes Project repository (The 1000 Genomes Project Consortium 2015) with an average coverage of 3×. We observed a similar compression ratio as with simulated data in Figure 3.6-A.

3.5.2 Runtime Analysis

To assess the runtime efficiency of SECRAM, we considered the performance of its six most important subprocedures: *transposition*, *inverse transposition*, *compression*, *decompression*, *encryption*, and *decryption*. We ran experiments on simulated data sets with a range of coverages and error rates. Figure 3.7-A shows the runtime breakdowns in percentages for different subprocedures relative to total runtime, across four experiments. One important observation is that encryption and decryption are not bottlenecks in the system, demonstrating that our implementation of security does not come at the cost of efficiency. When coverage and error rate are low, compression is slower than encryption and dominates the runtime; this is because there is less information to encrypt, but it still takes time to compress the nonsensitive information. When the error rate is higher, the encryption time surpasses compression time, mainly because of order-preserving encryption (OPE) on more positions, but the global runtime does not change much even for a high error rate (1%). The high coverage hinders (inverse) transposition more than

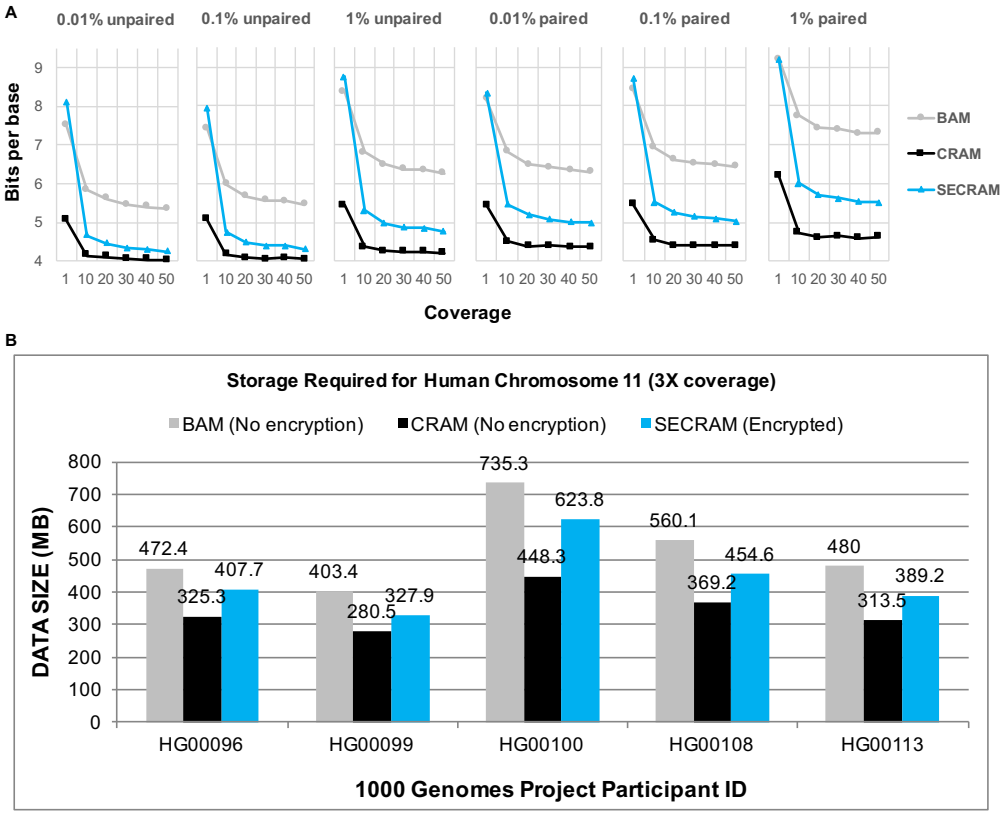

Figure 3.6: Storage analysis of compression algorithms for genomic data. (A) Storage comparison on simulated data sets. It shows the average number of bits per base (compression ratio) for three storage formats: BAM, CRAM, and SECRAM. The results are based on simulated data sets with different coverages (from one to 50) and error rates (0.01%, 0.1%, 1%) for both paired and unpaired data. (B) Storage comparison on Chromosome 11 from the 1000 Genomes Project participants with an average coverage of 3×. Only SECRAM is an encrypted format, whereas both BAM and CRAM are in plaintext. Both A and B show that the compression ratio of SECRAM is between that of BAM and that of CRAM.

compression and encryption because the complexity of (inverse) transposition is linearly dependent on the number of total bases. We also observed that decompression is the most efficient subprocedure; this is because the decompression dictionaries for some compression algorithms (e.g., Huffman encoding) are stored with SECRAM and can be directly used, whereas compression has to build these dictionaries.

Figure 3.7-B shows the runtime costs of conversion between BAM and SECRAM on real data sets as used in Figure 3.6-B. The conversion from BAM to SECRAM consists of three procedures (transposition, compression, and encryption), whereas the conversion from SECRAM to BAM encompasses the remaining three (inverse transposition, decompression, and decryption). On average, to process one megabyte of data, each procedure takes < 0.25 seconds; in total, the conversion in either direction takes < 0.5 seconds. We should emphasize that our current implementation is parallelizable (i.e., numerous data files or chromosomes can be processed in parallel). For instance, a whole-genome

BAM file of 100 gigabytes can be converted to SECRAM in parallel with 24 threads (one thread for each reference chromosome: 22 autosomes plus two sex chromosomes) in $\sim 1h$.

So far, we have analyzed the runtime costs of converting a whole BAM file or a whole SECRAM file, which is a one-time cost. For most applications, conversion is not necessary, but selective retrieval is crucial for efficiency. We designed this new format so that laboratories need only to store the inherently encrypted, compressed SECRAM files and develop analysis pipelines for accessing the data directly from these files. Therefore, it is critical to evaluate the efficiency of the SECRAM format for data retrieval. Figure 3.7-C shows the average runtime performance of random access of a data region of one megabase (the cost scales linearly with the size of the region). The total time for the retrieval procedure is < 0.5 seconds, which is efficient for real-time usage in any analysis pipeline. In traditional use, without any security measures, data retrieval consists of at least three steps (*random access*, *disk reading*, and *decompression*); decryption roughly doubles the runtime overhead, which we believe to be acceptable, given the strong privacy and security features.

3.5.3 Security and Privacy Analysis

Two encryption schemes are used in SECRAM. The symmetric encryption (SE) scheme used in SECRAM provides semantic security that is a standard and strong security guarantee in cryptography. OPE is less secure than SE (see Section 3.6). In SECRAM format, information is organized by genomic position. Therefore, during retrieval, the storage server can search the encrypted data (due to the properties of OPE) to locate the exact range of data in the query. The stream cipher mode enables the server to restrict the decryption key stream to only the information within the query range; hence, private information about other positions is protected from clients who send the query. This is in contrast to applying straightforward encryption techniques to read-based data, which risks information leakage from overlapping reads with each retrieval. Therefore, the position-based storage used in SECRAM is crucial for its fine-tuned data protection and privacy control.

3.5.4 Real Case Study

We envision SECRAM as most useful when deployed for routine clinical care. There are two phases that typically comprise the clinical application of genomic data: (1) variant calling: a hospital outsources a large volume of sequence data to the cloud and delegates variant annotation to a specialized bioinformatics company (e.g., Sophia Genetics [124]); and (2) mutation querying: a clinician in the hospital queries for hotspot mutations for precise diagnosis of a patient. In the first phase, the hospital should restrict the company's access to a predefined set of genes for variant calling in order to respect patient privacy, whereas in the second phase, a clinician should only access the genes that are related to their area of specialty. Both phases require the property of selective retrieval on encrypted sequence data on the cloud, ideally without having the hospital frequently handle access control on data requests. Typically, the data corresponding to a gene (or a set of genes) are accessed twice, necessarily in the first phase of variant calling and potentially in the second phase of mutation querying if the clinician requires verification of raw sequence data.

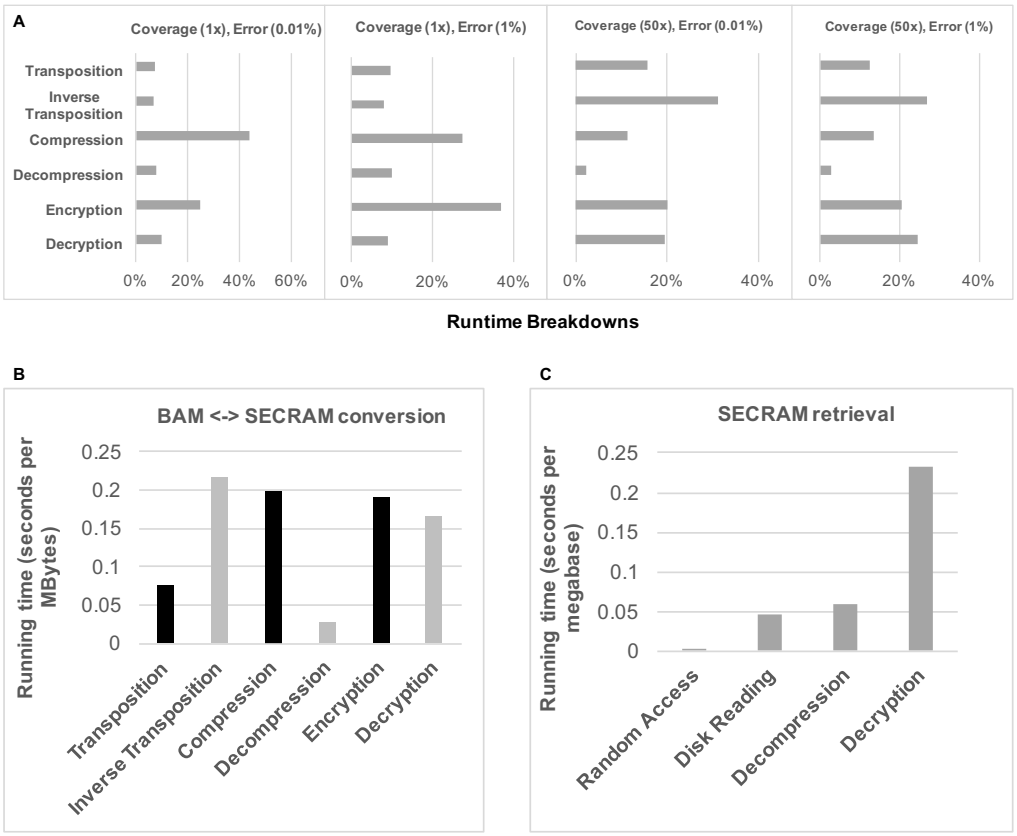

Figure 3.7: Runtime analysis of SECRAM system for storing and retrieving genomic data. (A) Runtime breakdowns on simulated data sets for the six most important procedures in SECRAM: transposition, inverse transposition, compression, decompression, encryption, and decryption, relative to total runtime (= 100%). The experiments were repeated for four simulated data sets with low (1×)/high (50×) coverages and low (0.01%)/high (1%) error rates. In all cases, the runtime cost of encryption/decryption is comparable with other necessary procedures, showing that enforcing security does not result in significant performance overhead. (B) Conversion time on real data sets (same data sets as those in Figure 3.6-B). It shows the average conversion time between the BAM and SECRAM formats on real data sets, running with a single thread on a machine equipped with Mac OS X Yosemite system and 3.1-GHz Intel Core i7 processor. The black bars (transposition, compression, encryption) represent the three steps of conversion from BAM to SECRAM, whereas the light gray bars (decryption, decompression, inverse transposition) represent the three steps of conversion from SECRAM to BAM. Each step takes < 0.25 seconds per megabyte of data. (C) Retrieval time on real data sets (same data sets as those in Figure 3.6-B). It shows the average runtime cost of retrieving data within a range of 1 million genomic positions. The actual data size corresponding to 1 million positions depends on the coverage. Shown are experiments on real data sets from Figure 3.6-B with a coverage of about 3× and a size of slightly > 1 megabyte.

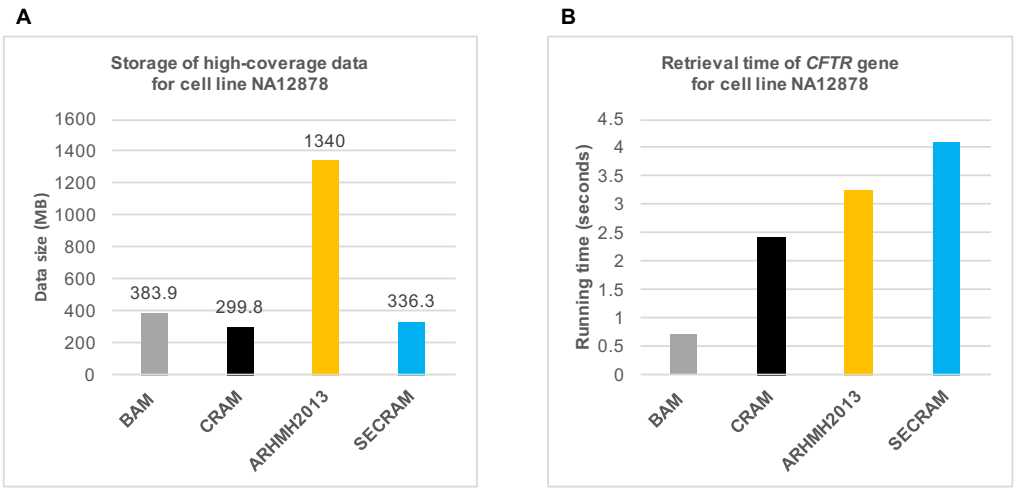

Figure 3.8: (A) Storage and (B) runtime on high-coverage clinical data. The data come from a public cell line (NA12878), based on a gene panel that includes the CFTR gene, queried for diagnosing cystic fibrosis. The average coverage of these data is 1035, containing more than 4 million reads of length around 300 bases. ARHMH2013 [115] is a privacy-preserving solution on BAM files that does not address the compression requirement. We observe a consistent compression performance of SECRAM on high-coverage clinical sequence data. Moreover, querying for the CFTR gene on SECRAM takes less than twice the time on the non-encrypted CRAM.

As a case study, we evaluate the storage and retrieval performance by using high-coverage sequence data in clinical care: specifically, the diagnosis of cystic fibrosis associated with the CFTR gene (roughly between positions 117120017 and 117308719 on Chromosome 7 of the reference sequence GRCh37). The sequence data come from a public cell line (NA12878), based on a gene panel that includes the CFTR gene. The maximum coverage is 10642×, whereas the average coverage is 1035×. In addition, we compare SECRAM with another solution that we call ARHMH2013 [115] that provides similar security and privacy properties for BAM files. In Figure 3.8-A, we observe a consistent compression performance of SECRAM compared with the low-coverage results in Figure 3.6; however, the ARHMH2013 approach does not scale well with high-coverage data due to the lack of compression and the overhead of encryption. In Figure 3.8-B, we also see that SECRAM performs very well in selective retrieval, namely, less than twice the time needed when using non-encrypted CRAM. We should emphasize that with SECRAM, one can query for a small region of data, while standard encryption approaches necessitate downloading the entire genome, making SECRAM more efficient as well as privacy-preserving.

3.6 Discussion

In this work, we present a novel, position-based, compressed, encrypted format for storing genomic data that enables selective and secure retrieval of variant information. We demonstrate that our solution compresses data more effectively than widely used formats like BAM. Importantly, it offers a level of security not possible with existing standard

storage formats, preventing the data leakage to which read-based formats are susceptible, without slowing down data retrieval or analysis. We provide algorithms for conversion to and from BAM/read-based formats, compression, decompression, encryption, and decryption.

3.6.1 Attacks Against OPE

As described below, the ciphertext of OPE contains both order and equality information about the underlying data. Therefore, if it is assumed that an attacker has enough information, OPE-encrypted data are vulnerable to some well-known attacks that exploit this property, such as frequency analysis attacks [120, 125, 126]. However, these frequency-analysis attacks would not be effective against our scheme, because (1) positions are encrypted only once regardless of the coverage, because all information about each position is clustered together in the transposition phase of SECRAM; (2) we use different OPE keys for data from different individuals; and (3) the underlying message in our adopted OPE scheme is usually a subset of the whole position space of genomic data, and it matches the requirement of the OPE scheme for message unpredictability. Regarding the third reason, however, the level of data protection depends on the data, more specifically, on the distribution of genetic variants and sequencing errors across the genome. If average coverage and sequencing error rate are high enough such that there is at least one different base from the reference genome on almost every position, SECRAM protection with OPE will be less effective. In this case, OPE would have to encrypt the whole plaintext space and thus provide little protection. The compromise of position information also leads to potential threats for the content encrypted by SE. For example, if positions are known and coverage can be inferred from the encrypted content size, CNV (copy number variation) analysis can identify deletions/insertions without the need for the sequence itself, and other single-nucleotide variants can be inferred by observing peaks of encrypted information size on the compromised positions. Nevertheless, a countermeasure would be to shuffle the OPE ciphertext [115], which is essentially a second-level encryption and reduces the efficiency of retrieval. An alternative countermeasure is to add random padding to each position to obfuscate the size of encrypted content at the cost of storage efficiency.

The purpose of adopting OPE as a building block is to maximize the difficulty of launching a successful attack against the encryption scheme while still allowing for efficient retrieval from the ciphertext. To enable efficient retrieval, the order information must be observable from the ciphertext, and under this premise, OPE is the optimal available solution with the best security guarantee, as the database community has recognized long ago [127]. We do not rule out the possibility that there could be other variants of frequency-analysis attacks against the OPE scheme adopted in this work; however, there already exist several enhanced OPE schemes [128, 129, 130] as potential replacements of the current version. Another alternative to OPE would be secure multiparty computation (SMC), for example, SMC schemes based on homomorphic encryption. Considering the high volume of data in the problem addressed here, however, SMC would introduce a prohibitively high overhead to storage and retrieval.

3.6.2 Lossy Compression

In the open-source implementation of CRAM [96], multiple lossy compression options are provided, such as removing read names and reducing the precision of quality scores. These options are also made available in our solution. For instance, we noticed that quality scores account for most of the storage space. We provide a parameter to specify the precision of quality scores; the default option maintains the original scores to allow exact reconstruction of the BAM file. A compression algorithm called Quartz [112] can be used to discard 95% of quality scores without affecting the accuracy of downstream analysis. Although their solution requires further validation, it indicates that lossy compression is a possibility for reducing storage requirements without jeopardizing the value of the data.

3.6.3 Datatype Applicability

SECRAM is designed to be compatible with SAM/BAM files; hence it enables, or is extensible to, a variety of sequence data that can be handled by SAM/BAM. These include (but are not limited to) the following:

- Multiple mapping. A read can be mapped to multiple alignments, for example, due to repetition. One of these alignments is considered primary, whereas the other alignments are annotated as secondary and have a link to the primary alignment. In BAM files, the read bases and quality scores of secondary alignments are set to "empty" to reduce the file size. SECRAM can be extended similarly to nullify bases and quality scores of the corresponding positions of secondary alignments and to include a link to the primary alignment in the read headers of these secondary alignments.

- Long reads. Several high-throughput sequencing systems produce long reads (e.g., the Sequel system [131], with read length 10 - 15 kb), for which SECRAMs prevention of read-based information leakage is even more valuable. Similar to CRAM, SECRAM is expected to have a slightly higher compression ratio with longer reads, assuming the same coverage depth, because there is less read information (e.g., read name). However, this improvement is modest because the majority of storage ($> 80\%$ in our data) is occupied by read bases and quality scores.

- RNA-seq data. RNA sequence data are stored in the same way as DNA sequence data in BAM formats and hence can be processed in essentially the same manner by SECRAM.

3.7 Summary

Overall, our solution addresses the pressing issues of data storage and security brought about by advances in sequencing technology and the emergence of personal and clinical genomics. By bridging the persistent gap between compression and security in the storage of genomic data, SECRAM offers an effective balance between the needs of researchers for efficient data analysis and the needs of individuals to maintain their genetic privacy. It will be important to continuously reevaluate the standards of genomic data storage as novel technologies are developed, security threats arise, and more complex

phenotypic analyses become possible. Integrative solutions that carefully consider the use and misuse of personal genomic data are essential for ensuring its secure, effective storage and maximizing its utility in treating and preventing disease.

Chapter 4

Protecting genomic data with arithmetic-computation capability

Apart from searching on protected genomic data, more sophisticated operations are required in many important use cases. For instance, researchers may want to issue statistical queries on encrypted genomic database, or doctors may want to compute disease risks by incorporating genomic information. There are various tools for executing sophisticated computation on protected genomic data, but each comes with certain drawbacks:

- *homomorphic encryption* and *secure multi-party computation*: High runtime overhead, communication cost, and storage blowup;
- *differential privacy*: Accuracy degradation;
- *trusted computing (e.g., Intel SGX)*: Questionable trust model.

As we mentioned, the searching functionality in the previous chapter could also be implemented with homomorphic encryption or secure multiparty computation, but it would be impractical, considering the huge amount of data. In this chapter, in order to perform arithmetic computation, we introduce a system based on secure multiparty computation, by focusing on an important use case, namely, data quality control in genomic applications.

Large human genomic databases enable researchers to conduct whole genome analysis on genomic variations and their associations with phenotypic traits. For instance, in a case-control *Genome-Wide Association Study* (GWAS) scenario, researchers can compute χ^2-statistics to reveal whether some genetic variants and a given disease are independent or not up to a significance level. In another case, researchers might use linear regression for modeling the relationship between a continuous phenotypic trait (dependent variable) and a genetic variant (explanatory variable). Although small-scale GWAS can identify variants associated with diseases, these variants explain only a small portion of the risk variability for many diseases. Discovery of new significant associations requires much larger sample sizes. However, the privacy-sensitive nature of human genomic data prevents research groups from sharing genomic data of individual participants, especially when it comes to cross-border collaboration. Therefore, it is usually

hard or even impossible to directly combine data from several groups to enlarge sample sizes. Nevertheless, to increase statistical power and reduce false-positive findings, researchers can use *Genome-Wide Association Meta-Analysis* (GWAMA) for the synthesis of results from multiple independent studies, thanks to the fact that GWAMA requires only aggregate statistics rather than individual genomic data from each study.

Generally speaking, GWAMA can be divided into two phases: the data-quality-control phase and the meta-analysis phase. Quality control (QC) usually takes place before meta-analysis, in order to filter out the studies whose data have quality issues and thus are not appropriate for meta-analysis. Both phases involve multiple processing steps of summary data from individual studies. Quality control is an important procedure for ensuring the quality or even the correctness of a meta-analysis.

Our proposed system is built upon a comprehensive protocol describing the state-of-the-art procedure to conduct data QC for large-scale GWAMA [132]. This particular framework has incorporated all the necessary QC steps proven to be helpful in practice. In this framework, each study shares its own association statistics for each genetic variant, such as allele frequencies, P-value and effect size estimates with standard errors, which are delivered to the analysts. The analysts will apply statistical techniques to detect potential errors in the aggregate data. For example, quality control will check whether the same allele is designated as the effect allele for all studies, in order to prevent that one study erroneously declares the other allele as the effect one. These quality issues could lead to unexpected or incorrect conclusions for the variant associations if the data were used blindly. We should emphasize that the quality control procedure is designed to detect systematic errors of a study, hence if such an error exists, many data points (i.e. statistics of single nucleotide variants (SNVs)) will deviate significantly from the expected ones, and thus this phenomenon can be easily observed by plotting the data.

However, even aggregate statistics may contain sensitive information from each participant; it has been demonstrated that individual information (e.g., cohort membership) can still be inferred from the published summary data [11, 133, 134, 135, 136, 137, 12]. As a matter of fact, the practice of publishing certain aggregate data has been discouraged since the revelation of the privacy breach of public summary statistics. For instance, the NIH[1] removed aggregate genomic data (e.g. allele frequencies and P-values) from open-access databases, and urged the scientific community to take precautions before sharing any aggregate GWAS data [138]. These statistical inference attacks are based on a similar idea: if an adversary has access to a large number of SNVs (both the victims SNVs and the aggregate results of the SNVs), it can gain strong enough statistical power to tell whether the victim is a contributor to the dataset. If a dataset contains sensitive health information about participants such as their HIV status, such an inference is a serious privacy breach.

The privacy issue of sharing plaintext aggregate statistics resides in that they reveal aggregate statistics for a high number of SNVs (we call these SNV-level statistics, formally defined in Section 4.1.1), and these statistics are exactly the input to a quality control procedure and a meta-analysis model. A secure approach is needed in order to complete the procedure without revealing the SNV-level statistics of any single study. A secure protocol for the meta-analysis phase has been proposed by operating on encrypted study statistics such that only the result of the meta-analysis is revealed [29]. However, as we shall see, quality control is a fundamentally more complicated procedure, and

[1] National Institutes of Health in United States

protecting this phase remains an open problem. In the proposed secure quality control (SQC), it guarantees that the analysts will receive nothing other than the final quality measurements. To the best of our knowledge, this is the first comprehensive solution for secure quality control for meta-analysis of genome-wide association studies.

4.1 Background

4.1.1 SNV-Level Aggregate Statistics in Quality Control

A meta-analysis protocol usually requires the aggregate association statistics such as effect allele frequency (EAF) (f), beta estimate[2] (β), standard error (e) and P-value (p), for a predefined set of variants. We denote the set of SNV-level statistics for study j as $\{(f, \beta, e, p)_{i,j}\}$, $i \in \{1, 2, \cdots, n\}$, where n is the number of variants. We use these four types of statistics for demonstrating our solution in a secure quality control pipeline, but it is easy to extend the solution to handle other types of statistics. We also use N_j to denote the sample size (number of participants) of study j, and N_j is usually a public value.

4.1.2 Secure Multi-Party Computation

A well-established cryptographic approach called secure multi-party computation (SMC) guarantees the following property (without loss of generality, we use two parties throughout the chapter):

Definition 4.1 (Secure two-party computation)**.** *Suppose there are two parties, Alice and Bob, who have their own private data x and y, respectively, and want to securely compute a function $F(x, y)$.The SMC technique enables Alice and Bob to execute a protocol that outputs the result $R = F(x, y)$, without Alice learning any other information on Bob's input y, nor Bob learning any other information on Alice's input x.*

Using this technique in the quality-control scenario, we can imagine splitting each statistics into two random and individually uninformative shares (secret shares), one stored by Alice and one stored by Bob. The quality control can then define a set of functions to be securely computed on the two shares. The underlying statistics are not revealed as long as Alice and Bob do not collude, which is realistic in practice if, for example, the two parties are chosen to be private, competitor companies (e.g., a Google cloud server and an Amazon cloud server). A detailed system setup of our solution is described later.

There are many cryptographic protocols for SMC, one of them being Yao's garbled circuit [139], which is what we use for the evaluation in this chapter. To summarize the technique, it can be decomposed into the following steps:

(i). The function $F(x, y)$ is described as a Boolean circuit with $\log x + \log y$ input gates. The circuit is known to both Alice and Bob.

(ii). Alice *garbles* (encrypts) the circuit. Hence Alice is called the *garbler*.

(iii). Alice sends the *garbled circuit* to Bob along with her encrypted input (encryption of x).

[2]Beta estimate is the coefficient learnt in linear regression model.

(iv). Bob receives his encrypted input (encryption of y) from Alice through *oblivious transfer* [140].

(v). Bob *evaluates* (decrypts) the circuit (NOT the inputs) and obtains the encrypted output. Hence Bob is called the *evaluator*.

(vi). Alice and Bob communicate to learn the output.

The above Step (iv) uses a protocol called oblivious transfer, whose functionality can be defined as follows:

Definition 4.2 (Oblivious transfer)**.** *Suppose there are two parties, Alice and Bob. Alice has two values v_0 and v_1, whereas Bob has one value $i \in \{0, 1\}$. Oblivious transfer (OT) enables Bob to obtain value v_i, without Alice learning i, nor Bob learning v_{1-i}.*

To make things more clear, we present below a simple (but inefficient) implementation of Yao's garbled circuit in Protocol 0. Researchers have proposed various optimization techniques to improve its efficiency, such as Free XOR gates [141], Half AND gates [142], OT extension [143, 144, 145] and Point-and-permute [146, 141, 147]. Discussing all these techniques is out of scope of this thesis, although most of them are implemented in the library used by our evaluation.

4.1.3 Differential Privacy

Informally, differential privacy guarantees that a persons contribution to a dataset does not significantly alter the output distribution of a statistical query on the dataset. Let $\mathcal{U}$ denote the universe of entities, $2^{\mathcal{U}}$ denote the powerset of $\mathcal{U}$, and D ($\in 2^{\mathcal{U}}$) denote a dataset.

Definition 4.3 (Differential privacy)**.** *A randomized mechanism $\mathcal{K}$ provides (ϵ, δ)-differential privacy if and only if, for any two data sets D and D' which differ in at most one individual, and for any $S \subseteq range(\mathcal{K})$,*

$$Pr[\mathcal{K}(D) \in S] \leq e^{\epsilon} Pr[\mathcal{K}(D') \in S] + \delta.$$

We then introduce the sensitivity of a function $f : 2^{\mathcal{U}} \rightarrow \mathbb{R}^d$: it characterizes the largest possible change in the value of f, when one data element is replaced.

Definition 4.4 (l_k-sensitivity)**.** *The l_k-sensitivity of a function $f : 2^{\mathcal{U}} \rightarrow \mathbb{R}^d$ is $\Delta_k f = max_{D_1, D_2} ||f(D_1) - f(D_2)||_k$, where D_1 and D_2 are datasets differing in a single element. k is usually 1 or 2.*

In the standard approach of adding Laplacian noise, we are concerned about l_1-sensitivity; releasing $f(D) + w$, where w is random noise drawn from a Laplace distribution with mean 0 and scale $\frac{\Delta_1 f}{\epsilon}$, satisfies $(\epsilon, 0)$-differential privacy, or just ϵ-differential privacy. For example, to release minor allele frequency of a SNV while satisfying ϵ-differential privacy, it is proved that we should add Laplacian noise with scale $\frac{\Delta_1 f}{\epsilon} = \frac{1/N}{\epsilon}$, where N is the size of the dataset [13]. By adding Gaussian noise with standard deviation that depends on ϵ and δ, we can achieve (ϵ, δ)-differential privacy [148], which is discussed in more detail in Section 4.3.1. Smaller ϵ and δ imply higher noise variance and stronger privacy. Although differential privacy reduces the accuracy of GWAS applications, we

Procotol 0 Garbled circuit protocol

- **Input**: Alice has input x, Bob has input y.
- **Circuit description**: Alice and Bob describe the function $F(x, y)$ as a circuit out of 2-input XOR and AND gates.
- **Garbling**: For any gate G (XOR or AND), let us denote its two input wires as w^a, w^b, and output wire as w^c. For each wire, Alice generates two 128-bit random values for the two possible values (0 or 1) of the wire. In other words, we have the following:

  - Wire w^a: Generates X_0^a and X_1^a;
  - Wire w^b: Generates X_0^b and X_1^b;
  - Wire w^c: Generates X_0^c and X_1^c.

  Afterwards, Alice garbles the truth table of G with a double-key symmetric encryption E_{k_1,k_2}. Namely, Alice generates four ciphertexts: $E_{X_i^a, X_j^b}(X_{G(i,j)}^c)$, for $i, j \in \{0, 1\}$. Moreover, Alice randomly permutes the four ciphertexts.
- **Data transfer**: There are three parts that require communication:

  - For each gate, Alice sends the corresponding four ciphertexts to Bob.
  - For Alice's input x with m bits $x_0, x_1, \cdots, x_{m-1}$ that correspond to the m input wires $w^{a_0}, w^{a_1}, \cdots, w^{a_{m-1}}$, Alice sends the corresponding m random values $X_{x_i}^{a_i}, \forall i \in 0, 1, \cdots, m-1$.
  - For Bob's input y with n bits $y_0, y_1, \cdots, y_{n-1}$ that correspond to the n input wires $w^{b_0}, w^{b_1}, \cdots, w^{b_{n-1}}$, Alice and Bob engage in n instances of 1-out-of-2 oblivious transfer protocols: in the i-th instance, Alice has two random values $X_0^{b_i}$ and $X_1^{b_i}$, Bob has the selection bit y_i, and finally Bob obtains its random value $X_{y_i}^{b_i}$.

- **Evaluation**: Bob evaluates the circuit from input to output. For each gate he encounters, he has four ciphertexts $E_{X_i^a, X_j^b}(X_{G(i,j)}^c)$ (for $i, j \in \{0, 1\}$), and two random input values X^a and X^b, where X^a is X_0^a or X_1^a, and X^b is X_0^b or X_1^b. Hence, Bob can decrypt exactly one of the four ciphertexts and reveal one output random value X^c. In the circuit, X^c might be used later (as an input random value $X^{a'}$ or $X^{b'}$) for the decryption of any following gate.
- **Revealing output**: After evaluating the whole circuit of function $F(x, y)$, Bob has a list of k output random values, $X^{c_i}, \forall i \in 0, 1, \cdots, k-1$, and Alice knows its mapping to Boolean values because she generated both random values $X_0^{c_i}$ and $X_1^{c_i}$ for every output wire w^{c_i}. Therefore, Alice and Bob can communicate to reveal the output $r = F(x, y)$, whose binary representation is: $r_i = 0$ if $X^{c_i} = X_0^{c_i}$, and $r_i = 1$ if $X^{c_i} = X_1^{c_i}$.

will justify its use in the case of quality control; in short, the way we apply differential privacy in quality control does not affect the utility of the data in the meta-analysis phase.

4.2 Adversary Model and System Structure

We adopt the standard security notion of semi-honest model for secure computation, which implies that corrupted parties do not deviate from the protocol specification but might try to gather information out of the protocol. From a data owner's point of view, potential adversaries include external attackers and curious or corrupted parties in the protocol (e.g., cloud servers or analysts in Figure 4.1). We assume the adversary knows the set of genomic variants SNVs $\{\text{SNV}_1, \text{SNV}_2, \cdots, \text{SNV}_n\}$ shared by all studies.

The storage and computation are outsourced to two non-colluding, semi-honest cloud providers (A and B). Each cloud server could be equipped with a cluster of machines to enable parallel secure computation. The system works as follows (Figure 4.1):

Step 1. Each study splits its aggregate statistics X (e.g. $f_{i,j}$) into two secret shares: $X = X_1 \oplus X_2$ (bit-by-bit XOR operation on the whole value), where X_1 is randomly chosen. X_1 is encrypted and sent to cloud A, whereas X_2 is encrypted and sent to cloud B. '$[X]$' represents the encryption of 'X' in Figure 4.1. Essentially, in our system, all the communication links are protected with authenticated symmetric encryption that is a well-established security feature in the Internet (e.g. when we use HTTPS instead of HTTP). The protection of X_1 in cloud A and X_2 in cloud B will be detailed in Section 4.5.

Step 2. An analyst sends a request to cloud A for checking data quality.

Step 3. The two clouds execute a secure two-party computation protocol to perform the quality control procedure on the secret shared input data of each study. The quality measurements are sent back to the analyst.

Step 4. If there is no quality issue, the analyst sends another request to cloud A for running a meta-analysis approach, and Step 5 follows. Otherwise, the analyst contacts the study whose data present quality issues, and the study administrator resolves these issues and goes back to Step 1.

Step 5. The two clouds execute another different secure two-party computation protocol to perform meta-analysis on the data. The result is sent back to the analyst. This step has been studied in previous work [29] and is not described here.

4.3 Secure Quality Control

In this section, we introduce our design for a *secure quality control* (SQC) procedure, and apply the design to several important quality-control scenarios. We divide this section into two sub sections: the first sub section describes several secure-computation procedures that serve as the building blocks for protocols in the second sub section.

4.3.1 Secure Computation Procedures

SQC is composed of several important procedures, each of which has an added value to the protection of the data. Unless otherwise specified, all the following procedures are performed in secure two-party computation by each of the two clouds on two secret

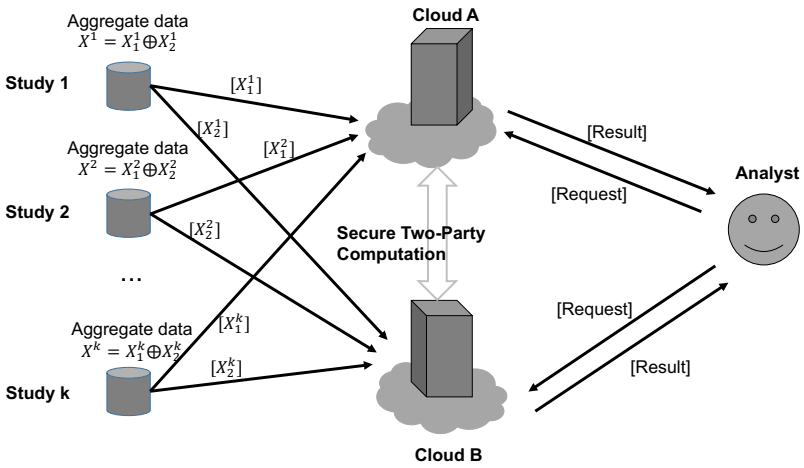

Figure 4.1: System architecture. Each study splits its summary statistics into two secret shares that are sent to two non-colluding cloud servers (e.g., Google and Amazon). At the request of an analyst, the two clouds perform secure two-party computation to verify data quality or perform meta-analysis. In both cases, only the final results, either quality measurements or meta-analysis results, are revealed to the analyst. '[X]' represents the encryption of 'X'. All the communication links are protected with authenticated symmetric encryption that is a well-established security feature in the Internet (e.g., HTTPS instead of HTTP).

shares of the input, which corresponds to Step 3 of the above system workflow. This is further explained by the general conversion guideline in Figure 4.2.

Procedure 1: Variant hiding by oblivious sorting

As discussed before, the adversary knows the list of targeted SNVs in the study, which is also an assumption for the aforementioned inference attacks. On the other hand, in the quality control protocols considered in this chapter, a set of SNV-level statistics will produce the same results regardless of the ordering among them, therefore we obliviously sort the statistics of the SNVs. In fact, this step hides the original ordering (e.g., based on positions on genome), which has already been used by previous work on meta analysis [149]. Procedure 1 serves as a preliminary step of SQC. The two clouds use bitonic sorting [150] to guarantee the obliviousness: the algorithm does not reveal information about the input [151]. In more detail, bitonic sorting works in a sequence of stages (number of stages depends on number of values to be sorted): in each stage, the algorithm compares *predefined* pairs of values in the list, and swap them if necessary. These *predefined* pairs are fixed for all lists having the same length, regardless of the values, which is why the sorting procedure is *oblivious*.

NOTE: Although we abbreviate it in the procedure, the input and output are actually represented as two secret shares. In other words, $L = L_A \oplus L_B$ and $L' = L'_A \oplus L'_B$, where L_A and L'_A are on cloud A, L_B and L'_B are on cloud B, and '$\oplus$' is a convenient abbreviation for the element-wise XOR between two list of elements. Moreover, the computation in the procedure (i.e., '$L' \leftarrow bitonicSort(L)$' in Procedure 1) is programmed as two-party computation circuits between cloud A and cloud B, each of which holds a

Plaintext Procedure

Require: X
Ensure: $f(X)$
$r \leftarrow f(X)$
return r

SMC Procedure

Require: Alice's input is X_1, Bob's input is X_2, and $X_1 \oplus X_2 = X$
Ensure: Alice and Bob output $f(X)$
Define $F(X_1, X_2)$ as follows:
$X \leftarrow X_1 \oplus X_2$;
$f(X)$.
Alice and Bob execute Protocol 0 for evaluating function F with Alice's input $x = X_1$ and Bob's input $y = X_2$. Alice and Bob reveal output $r = F(x, y)$ at the end of the protocol.
return r

Figure 4.2: SMC conversion. The figure shows a general guideline for converting a plaintext procedure to an SMC procedure. It applies to all procedures and protocols described in this section.

Procedure 1 Oblivious sorting

Require: : A list L
Ensure: : A sorted list L'

$L' \leftarrow bitonicSort(L)$
return L'.

share of the input (L_A and L_B respectively), rather than a traditional one that is executed on a single party that has the whole input. We stick to such an abbreviated presentation in all following procedures and protocols.

Procedure 2: Adding Gaussian noise

In most cases of genomic computation (e.g., GWAS, risk test), accuracy is of paramount importance. Therefore, many privacy-preserving techniques that introduce noise to data are criticized because of their unacceptable negative impact on computation accuracy [152, 24], even though a lot of advances have been made to apply these techniques in order to enable genomic data sharing [13, 14, 153, 16, 154]. However, quality control is meant to detect systematic errors of a study, hence analysts are concerned mainly about the global characteristics of the data, rather than the precise value of individual data points. For example, analysts might expect to see the data points cluster along the diagonal line of a pane rather than a horizontal line. Such a relatively loose requirement provides us room to change the data points of the results in a manner such that the output patterns can still lead to the same conclusion on the data quality (some results

are presented in Figure 4.7 and Figure 4.8).

In two-party computation, two methods have been explored for adding Laplacian noises to the output. In the first approach [155, 156], Laplacian noises are generated from an approximate Laplacian distribution. In theory, a Laplacian distribution can be exactly constructed from a uniform distribution because of the following proposition:

Proposition 4.1 ([157]). *If $X \sim U(-\frac{1}{2}, \frac{1}{2})$ (uniform distribution), then*

$$Y = \mu - b \cdot sgn(X) \cdot \ln(1 - 2|X|) \sim Laplace(\mu, b),$$

where μ and b are the mean and scale parameters of the Laplace distribution, and $sgn(X)$ denotes the sign of X.

This is a direct result of the inverse transform sampling. It is easy for two parties to jointly draw a sample from a uniform distribution without either party knowing the sample, as the summation of two uniformly distributed variable from same sample space remains a uniformly distributed variable in the sample space. However, this approach requires computing a logarithm operation afterwards, which can only be approximated through Taylor series with state-of-the-art SMC techniques and hence is computationally intensive. For example, it took about 5 seconds for two parties to generate a Laplacian scalar variable securely in a collaborative fashion on 2.4 GHz processors [156]. This remains impractical given the fact that we need to process millions of SNVs for each study.

In the second approach [158], a Laplacian sample is generated by using the following proposition:

Proposition 4.2 ([159]). *Let $X_i \sim \mathcal{N}(0, \lambda)$ for $i \in \{1, 2, 3, 4\}$ be four independent Gaussian random variables. Then,*

$$Y = (X_1^2 - X_2^2) + (X_3^2 - X_4^2) \sim Laplace(0, 2\lambda^2).$$

This proposition provides a way for two parties to cooperatively generate a Laplacian random variable that is unknown by each party. In this method, one party produces its noisy share $X_1^2 - X_2^2$, whereas the other one produces its noisy share $X_3^2 - X_4^2$. The two parties then engage in a secure addition operation, which is quite efficient. However, the problem with this approach is that the noisy shares, $X_1^2 - X_2^2$ and $X_3^2 - X_4^2$, do not follow the Laplacian distribution. Therefore, the method provides some privacy protection against each party, but it is not proven to satisfy any bound of differential privacy.

In this framework, we achieve differential privacy through adding Gaussian noise. We make use of the following two results.

Theorem 4.1 (Gaussian Mechanism [148]). *Let $\epsilon \in (0, 1)$ be arbitrary. For $c^2 > 2\ln(\frac{1.25}{\delta})$, the Gaussian Mechanism that adds zero-mean Gaussian noise with standard deviation $\sigma \geq \frac{c\Delta_2(f)}{\epsilon} > \frac{\sqrt{2\ln(\frac{1.25}{\delta})}\Delta_2(f)}{\epsilon}$ is (ϵ, δ)-differentially private.*

Proposition 4.3 ([160]). *Let $X_1, X_2 \sim \mathcal{N}(0, \sigma^2)$ be two independent Gaussian random variables. Then,*

$$Y = X_1 + X_2 \sim \mathcal{N}(0, 2\sigma^2).$$

Therefore, each party independently draws a random sample from the Gaussian distribution, computes the addition of the two samples via SMC, and adds the result to the statistics (Procedure 2). Because of Proposition 4.3, we can provide theoretical guarantee that the noisy result is differentially private against the two parties that engage in SMC and any external party that knows neither X_1 nor X_2. We have the following result.

Corollary 4.1. *Procedure 2 is (ϵ, δ)-differentially private against the honest-but-curious adversary which controls at most one of the cloud servers.*

Proof. Because an honest-but-curious adversary controlling Cloud A (resp., Cloud B) can remove its own noise X_1 (resp., X_2) from the output $f + w$, resulting in $f + w - X_1$ (resp., $f + w - X_2$), from Theorem 4.1, the output is (ϵ, δ)-differentially private. ■

Note that this procedure is executed on the fly for checking data quality without affecting the stored data, hence the data can still be used at a later stage (e.g., for meta-analysis) **without any accuracy loss.**

Procedure 2 Adding Gaussian noise

Require: : Original value f
Privacy parameters ϵ and δ
Sensitivity $\Delta_2(f)$
Ensure: : $f + w$, where w is a Gaussian noise

$\sigma \leftarrow \frac{\sqrt{2\ln(\frac{1.25}{\delta})}\Delta_2(f)}{\epsilon}$ (Theorem 4.1)
$X_1 \leftarrow \mathcal{N}(0, \sigma^2)$ (Cloud A draws a random sample X_1 from the Gaussian distribution)
$X_2 \leftarrow \mathcal{N}(0, \sigma^2)$ (Cloud B draws a random sample X_2 from the Gaussian distribution)
$w \leftarrow X_1 + X_2$
$r \leftarrow f + w$
return r

Procedure 3: Precision truncation

Truncating the precision of statistics serves two purposes in this framework: First, apart from differential privacy, it is an alternative way to add noise to the data; second, it helps to reduce the number of SNVs that can be revealed, undermining the power of inference attacks. The second feature will be discussed in the next procedure. In Procedure 3, we use fix-point representations (more efficient than floating-point operations), reserve a predefined precision and round the statistics to the nearest reduced-precision representation. For a fix-point statistics x that has w-bit width (total number of bits) and t-bit offset (number of bits for the fractional part), we preserve r bits after leading zeros (so that small fractional numbers, such as p-value 10^{-8}, are *not* truncated to zero).

Procedure 4: Oblivious de-duplication

After reducing precision, we might observe a high number of repetitive statistics. For example, if we have 4 million SNVs and preserve 10-bit precision (around 3 decimal digits), we could expect on average 4000 duplicates per SNV statistics. Not only is this frequency information useless for analysts to construct quality-control plots, but

Procedure 3 Precision reduction

Require: : Precision r bits, and a fix-point number x with w-bit width and t-bit offset.
Ensure: : A fix-point number x' that sets all other bits of x to zero except the first r bits after the leading zeros of x.

$x' \leftarrow x$
[Step 1] Starting from the most significant bit, skip leading zeros of x'
[Step 2] Preserve at most r bits after leading zeros
if there are m ($>= 1$) bits after Step 2 **then**
  $carry \leftarrow$ the most significant bit of the m bits
  Set the remaining m bits of x' to 0
  {We should avoid overflow when rounding up.}
  if the highest $(w - m)$ bits of x' are not all 1s **then**
    $x' \leftarrow x' + carry \times 2^m$
return x'.

also a potential privacy leakage to adversaries. In Procedure 4, we perform an oblivious sorting to cluster duplicated items together, preserve one of them by setting the others to the infinity element, and perform another round of oblivious sorting in order to discard the infinity elements. The second round of sorting is necessary to avoid leaking which elements are duplicated (and how many times they are duplicated).

Procedure 4 Oblivious de-duplication

Require: : a sorted sequence L of length n.
Ensure: : a de-duplicated sequence L' of length $n', n' \ll n$.

Apply Procedure 1 to L. We denote the sorted sequence as L'.
$previous\_item \leftarrow L'_1$
for $i = 2 \rightarrow n$ **do**
  if $L'_i == previous\_item$ **then**
    $L'_i \leftarrow \perp$ ($\perp$ is defined as the infinity element)
  else
    $previous\_item \leftarrow L'_i$
Apply Procedure 1 to L', and discard all the elements of $\perp$ at the end, resulting in a new sequence L'' of length n'.
return L''.

4.3.2 Secure Quality Control Protocols

Using the above design, we propose our methodology to run several quality-control protocols [132] in a secure and privacy-preserving way. In absence of the secure protocols described below, a large number of original SNV-level statistics would be revealed, making them vulnerable to inference attacks, hence each SQC protocol is necessary to avoid leaking those (precise) SNV-level statistics.

Protocol 1: Secure SE-N plot

An analyst can plot the median standard error (SE) across all SNVs against the square root of the sample size (N) for each study, in order to identify analytical problems,

e.g., inconsistent regression models. More specifically, for study j with sample size N_j (public value), the analyst finds the median in the corresponding list of standard errors: $e_{1,j}, e_{2,j}, \cdots, e_{n,j}$. We denote this median with $e_{median,j}$. If the study data have no quality issues, $\sqrt{N_j}$ is proportional to the inverse of $e_{median,j}$ by a public constant c [132]:

$$\sqrt{N_j} \approx c \cdot \frac{1}{e_{median,j}}.$$

Therefore, the study-specific data points of the SE-N plot should resemble a straight line. Protocol 1 applies oblivious sorting to each list of standard errors and chooses the corresponding median afterwards. A more efficient median-finding algorithm can be derived by using the selection networks [161, 151]. We note that there is no need to go through Procedure 3 and Procedure 4 in this protocol because the output for each study contains only one overall statistics (median) across all SNVs, which is different from the following two protocols.

Procotol 1 Secure SE-N processing

Require: :
List of standard errors from m studies: $L^1, L^2, \cdots, L^m$.
$L^j = \{e_{1,j}, e_{2,j}, \cdots, e_{n,j}\}$.
Ensure: :
List of median standard errors:
$\{e_{median,1}, e_{median,2}, \cdots, e_{median,m}\}$

for $j = 1 \rightarrow m$ **do**
Apply Procedure 1 to list L^j
$e_{median,j} \leftarrow$ the median of list L^j
return $\{e_{median,1}, e_{median,2}, \cdots, e_{median,m}\}$.

Protocol 2: Secure EAF plot

The EAF (effect allele frequency) protocol plots the reported EAFs from a study against another study, which could pinpoint issues such as a wrong strand, allele miscoding, etc. For example, a study might consistently label the wrong allele as the effect allele, leading to wrong allele frequencies. If the two studies j_1 and j_2 have no quality issues and their samples come from the same ancestry, then f_{i,j_1} should be close to f_{i,j_2}, $\forall i \in \{1, 2, \cdots, n\}$. The two sets of allele frequencies are taken as input to Protocol 2, which first reduces the precision of allele frequencies (Procedure 3) and then de-duplicates the pairs of EAFs (Procedure 4) from the two studies.

Protocol 3: Secure P-Z plot

The P-Z plot (P-value and Z-statistics) can reveal analytical problems of the computation of beta estimates (linear regression coefficients), standard errors or P-values. More specifically, for SNV_i of study j, the Z-statistics is computed as follows:

$$Z\text{-statistics} = \frac{\beta_{i,j}}{e_{i,j}}.$$

The corresponding reported P-value $p_{i,j}$ should be close to the P-value associated with the above Z-statistics. In the first step of Protocol 3, the two servers perform a secure

Procotol 2 Secure EAF processing

Require: :
List of effect allele frequencies from study j_1: $\{f_{1,j_1}, f_{2,j_1}, \cdots, f_{n,j_1}\}$;
List of effect allele frequencies from study j_2: $\{f_{1,j_2}, f_{2,j_2}, \cdots, f_{n,j_2}\}$;
Precision: r bits.
Ensure: :
List of reduced-precision and de-duplicated EAF tuples:
$\{(f''_{i,j_1}, f''_{i,j_2})\}$, $i \in \{1, 2, \cdots, n'\}$, $n' \ll n$.

for $i = 1 \rightarrow n$ **do**
$f'_{i,j_1} \leftarrow$ Procedure 2 and 3 on number f_{i,j_1}
$f'_{i,j_2} \leftarrow$ Procedure 2 and 3 on number f_{i,j_2}
return the result of applying Procedure 4 on the list of tuples:
$\{(f'_{i,j_1}, f'_{i,j_2})\}$, $i \in \{1, 2, \cdots, n\}$.

division protocol to compute the Z-statistics, which is known to be a costly operation in secure computation. We benchmark this step in Section 4.4.2. The resultant Z-statistics are paired with the corresponding P-values to go through a similar precision-reduction and de-duplication procedure as in Protocol 2.

Procotol 3 Secure P-Z processing

Require: :
Lists (length n) of p-values, beta estimates, and standard errors from study j;
Precision: r bits.
Ensure: :
List of reduced-precision and de-duplicated P-Z tuples:
$\{(p''_{i,j}, Z''_{i,j})\}$, $i \in \{1, 2, \cdots, n'\}$, $n' \ll n$.

for $i = 1 \rightarrow n$ **do**
$Z_{i,j} \leftarrow \frac{\beta_{i,j}}{e_{i,j}}$ (secure division in garbled circuits)
for $i = 1 \rightarrow n$ **do**
$p'_{i,j} \leftarrow$ Procedure 3 on precision r and number $p_{i,j}$
$Z'_{i,j} \leftarrow$ Procedure 3 on precision r and number $Z_{i,j}$
return the result of applying Procedure 4 on the list of tuples:
$\{(p'_{i,j}, Z'_{i,j})\}$, $i \in \{1, 2, \cdots, n\}$.

4.4 Implementation and Evaluation

We use ObliVM-GC [162] for our backend secure two-party computation. This framework provides a Java library that helps programmers to build secure-computation protocols in a circuit representation. Nayak et al. wrap it into a parallel secure-computing paradigm that we use to scale up our solution [163]. Our experiment uses two sets of data:

- Real data: GWAS data from 10 studies, each of which builds a linear regression model between human height and each of roughly three million SNVs [164].
- Simulated data: Following another work [165], we choose a study size (N) and a number of SNVs (n). For each SNV, we pick a random number in the range of

0.05 to 0.5 as the minor allele frequency. We then generated the genotypes of N individuals independently. For example, we choose $N = 1000$ and $n = 3000000$, which is similar to the size of our real data.

4.4.1 Privacy and Utility Analysis

Log likelihood ratio test. As we apply differential privacy to the allele frequencies that are the major vulnerability exploited by inference attacks, we experimentally analyze the privacy of our solution by using the attack power as a metric. The log likelihood ratio test is one of the most powerful inference attacks for individual detection in a pool [136]. To construct the test, we assume a reference population with specified allele frequencies for variants, and we sample a pool of 1000 individuals from this reference population. The null hypothesis of the test is that a tested individual is *not* in the pool. An LLR statistic for the tested individual is computed based on the genotype frequencies of the reference population and the pool, in order to tell whether the individual is included in the pool.

The larger the LLR statistic is, the higher is the probability that individual i is in the case group (i.e. lower P-value for the test). To assess the attack power, we construct two groups: one case group (same as the pool) and one test group, for nine different experiments. We then calculate the LLR statistics for each individual in the two groups, rescale them to the range $[0, 1]$, and analyze their distribution (Figure 4.3). A wider gap between the case distribution and the test distribution indicates a higher attack power. From the results, it is evident that privacy can be enhanced by requiring a higher level of differential privacy and releasing fewer SNVs.

Note that the absolute values of the LLR statistics are invisible in Figure 4.3, due to the rescaling operation; in fact, for the eight experiments of adding noise, the LLR statistics are extremely small (hence P-values close to 1). This is because the effect of adding noise, even with the least amount of differential privacy ($\epsilon = 0.1, \delta = 0.05$), dominates over the effect of sampling, which indicates that the tested individual is actually closer to the reference population than to the 'noisy' pool. However, we can still observe the interesting relative difference between the case group and the test group. To quantify the attack power, we measure the gap between the case distribution and the test distribution of Figure 4.3 by using Kolmogorov-Smirnov distance (instead of performing a Kolmogorov-Smirnov test to tell whether the two samples are drawn from the same distribution, we are more interested in measuring the distance between the two). The result is illustrated in Figure 4.4. A larger distance indicates less privacy, because it would be easier for an attacker to differentiate the two groups. We observe that adding the least amount of differential privacy in SQC can already greatly reduce the attack power and strengthen the privacy.

Most of the time, an attacker does not have access to genotypes of a large group of individuals. But assuming the attacker has the genotype of a victim, he must make an assessment about whether the victim is in the pool based on one LLR statistic. Note that using the P-value for the assessment is not effective in our scenario for the same reason discussed above. From an attacker's point of view, Chen et al. define a classifier with a threshold τ: if $LLR(i) \geq \tau$, then individual i is classified to be in the case group, otherwise it is in the test group [166]. To have a high true positive rate, we set a low τ value, e.g. at the lowest 1 percentile of the case group (99% true positive rate).

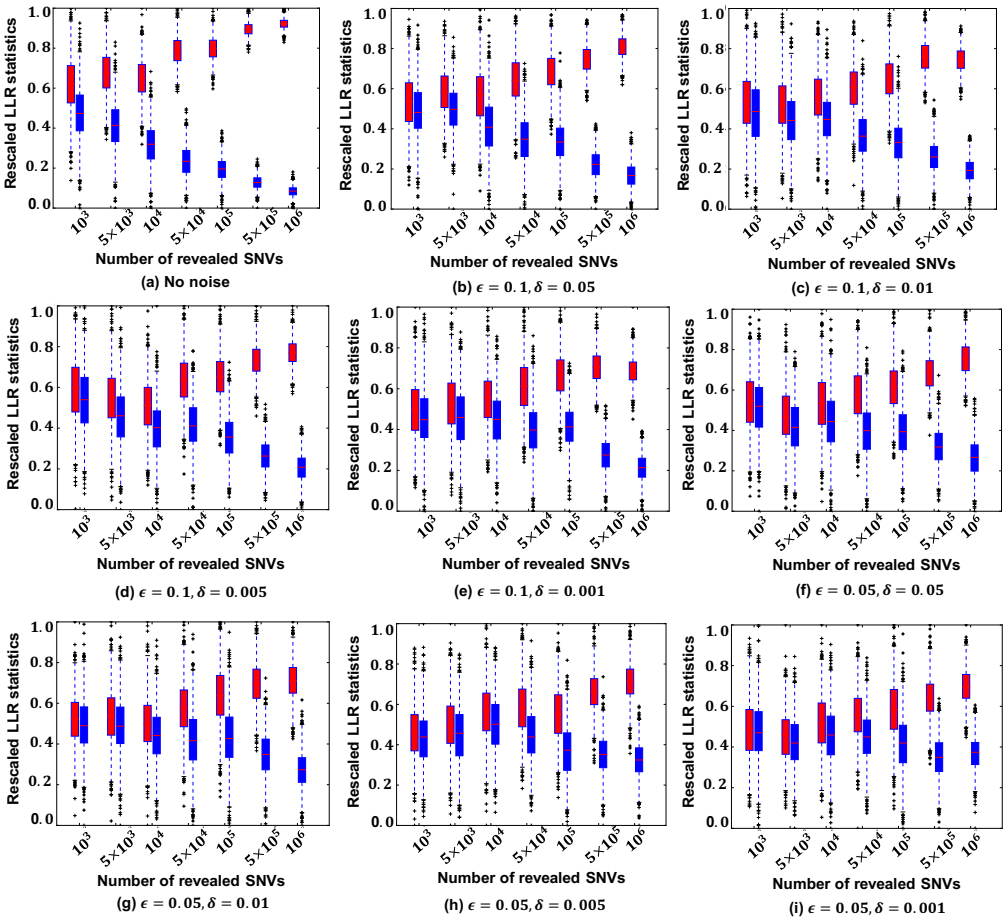

Figure 4.3: Rescaled LLR statistics. The nine experiments are conducted on simulated datasets after adding different Gaussian noises by choosing parameters ϵ and δ. In each experiment, for the case group (red) and the test group (blue), we reveal different number of SNVs, from left to right: 1000, 5000, 10000, 50000, 100000, 500000, 1000000. In general, we observe that revealing more SNVs makes it easier to separate the two groups based on LLR statistics. Adding Gaussian noises by imposing smaller values of ϵ and δ helps to mitigate the effect of this attack. We could choose these parameters by basing them on the quantification of the resulting attack powers, which will be detailed in Figure 4.4 and Figure 4.5: for example, releasing 1000 SNPs after adding the least amount of differential privacy ($\epsilon = 0.1, \delta = 0.05$) is appropriate because the attack power is relatively weak in this case.

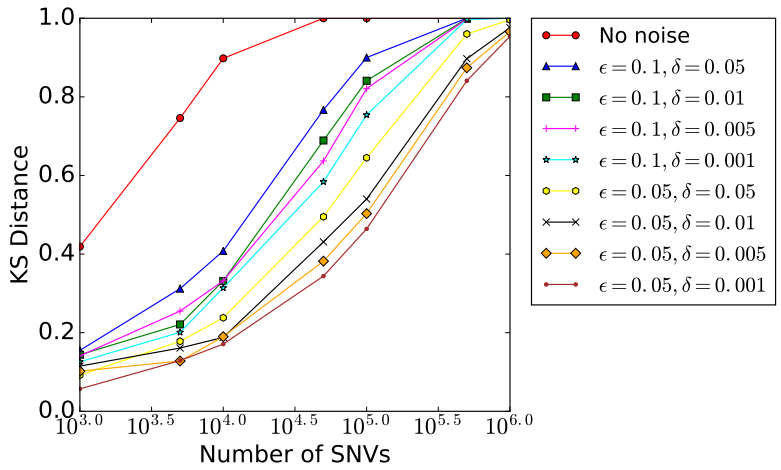

Figure 4.4: Kolmogorov-Smirnov distances between case LLR statistics and test LLR statistics. The distances correspond to the visualized gaps in Figure 4.3. The larger the distance is, the stronger the attacker power is, hence the less the privacy is. Adding the least amount of differential privacy ($\epsilon = 0.1, \delta = 0.05$) in SQC can already greatly reduce the attack power and strengthen the privacy.

We observe the false positive rates in the test group for the above nine experiments (Figure 4.5). If the data is released without adding noise, an attacker can easily achieve low false positive rates (i.e. effective attack) even if only a small number of SNVs are released. For example, the false positive rate is less than 40% when only 10000 SNVs are released. Differential privacy adding the lowest amount of noise in our experiment (i.e., $\epsilon = 0.1, \delta = 0.05$) can effectively thwart the attack. With stronger differential privacy, more SNVs can be released at the same false positive rate for an attacker. Nevertheless, we will see that with SQC, only a small number of SNVs need to be released in order to achieve good utility, hence we can add a low level of Gaussian noise for differential privacy, e.g., $\epsilon = 0.1, \delta = 0.05$.

Effects of precision r. From the above results, we observe that the number of SNVs used in the attack significantly influences the attack power. For example, when releasing one million SNVs, there is almost no privacy at all for all the privacy parameters in the experiments; but the strongest privacy parameters ($\epsilon = 0.05, \delta = 0.001$) that we use already have a non-trivial negative impact on the utility of the data (Figure 4.6). Therefore, it is crucial that our SQC protocols output a relatively small number of SNVs after de-duplication, hence achieving a good trade-off between privacy and utility. This number actually depends on the precision r in Procedure 3 used by Protocol 2 (analogous to Protocol 3): The higher the precision is, the more EAF pairs the protocol outputs. Roughly speaking, the maximum number of de-duplicated EAF pairs for precision r is 4^r .

We evaluate the effectiveness of our approach by setting different combinations of parameters (ϵ, δ, r), and observe the patterns of EAF plots in Figure 4.7. We also calculate the false positive rates (at 0.99 true positive rate) of LLR test on the output EAF data by assuming, pessimistically, that the adversary is able to link the EAF data back to

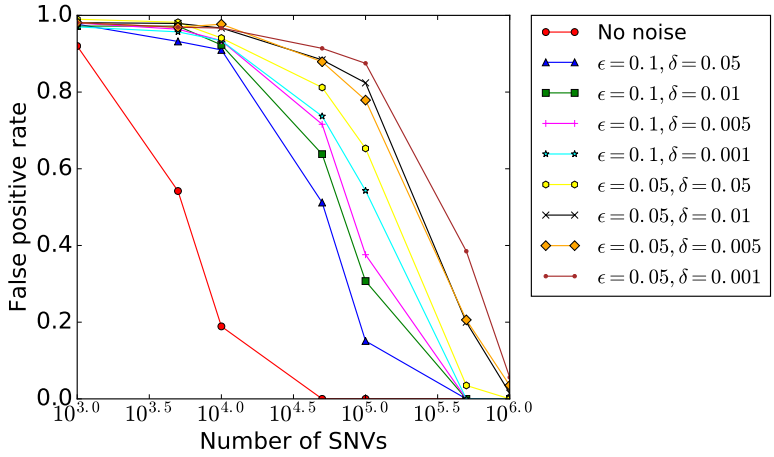

Figure 4.5: False positive rates of inferring an individual in the case group when setting the true positive rate to be 99%. The nine plots correspond to the experiments in Figure 4.3.

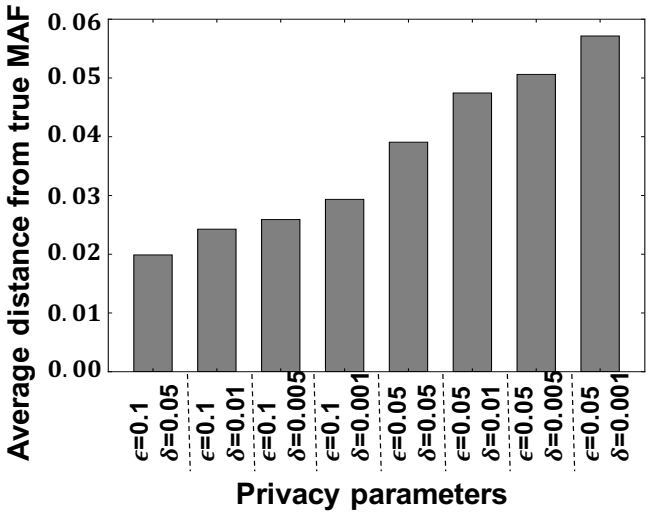

Figure 4.6: Average distance from original MAF after adding different Gaussian noises.

their SNV identifiers and then perform LLR test. As the precision increases, the number of de-duplicated EAF pairs grows exponentially and thus the false positive rate drops accordingly, but the plot is better preserved. Nevertheless, we can still observe the plot pattern even at a low precision $r = 5$, but it fades out when we further decrease the precision because points become sparse. By observing the distinction between Figure 4.7-b and Figure 4.7-c, the differential privacy parameter ϵ has a significant influence on the plot pattern, and $\epsilon = 0.1$ is a reasonable choice; in contrast, parameter δ has a relatively modest effect on the pattern. To guarantee a reasonable privacy level (i.e. high false positive rate) and preserve useful plot patterns, it is a good choice to set $\epsilon = 0.1$ and $r = 5$ or 7, and to fine tune the result by choosing δ.

Example: Visualization of EAF Plots

Due to Gaussian noises, precision reduction, and de-duplication, the numerical output of SQC protocols is very different from that of insecure QC protocols. However, in this section, we show that by guaranteeing a reasonable privacy level, SQC can still maintain the useful global characteristics for quality control. We take the secure EAF protocol as an example, because it encompasses all the above three procedures and results in the most noisy output. In our experiment, we set $\epsilon = 0.1, \delta = 0.05$, and precision $r = 7$. Combining these parameters, the secure EAF protocol outputs on average 4340 noisy and low-precision SNV pairs. Even if we assume that an adversary is able to link these SNVs back to their identifiers, it still has more than 90% false positive rate when trying to perform Homer's attack (Figure 4.5). We evaluate the utility of the output plots in Figure 4.8. It is evident that SQC maintains the quality (good or bad) of the data, with a slight dispersion of data points.

4.4.2 Runtime Analysis

Single machine. Secure computation provides strong security, but this comes at the cost of a high computational overhead. Even with current techniques and implementation, the slowdown of secure computation is as high as thousands or even hundreds of thousands of times, compared with non-secure, plaintext computation [162]. The following results are obtained on a single machine with Intel Core i7 processor clocked at 3.1 GHz and 16 GB of RAM.

Secure operation benchmark. Before executing the whole SQC protocols, we benchmark the different core computation procedures on a small set of 1000 SNVs. Table 4.1 shows the result of benchmarking. In secure computation based on garbled circuits, another measurement of performance is the number of AND gates generated during computation; this number is linearly proportional to the running time and communication cost. Oblivious sorting has a complexity of $O(n \log^2 n)$ [150], whereas all other procedures have a linear complexity as the number of SNVs increases. For example, with sequential implementation, we estimate it will take around 7.5 hours to accomplish oblivious sorting of 3 million SNV statistics of 64-bit representation. We also see that secure division incurs a high overhead, especially when bit length of numerator (denominator) increases; indeed, the complexity of a secure division is $O(l^2)$ for two numbers of l bits. In our SQC framework, secure division is only used for beta estimates and standard errors that are represented by 32-bit fix-point numbers, hence it will take 5.46 seconds on average to execute secure division for 1000 SNVs. This is roughly the same cost as oblivious sorting

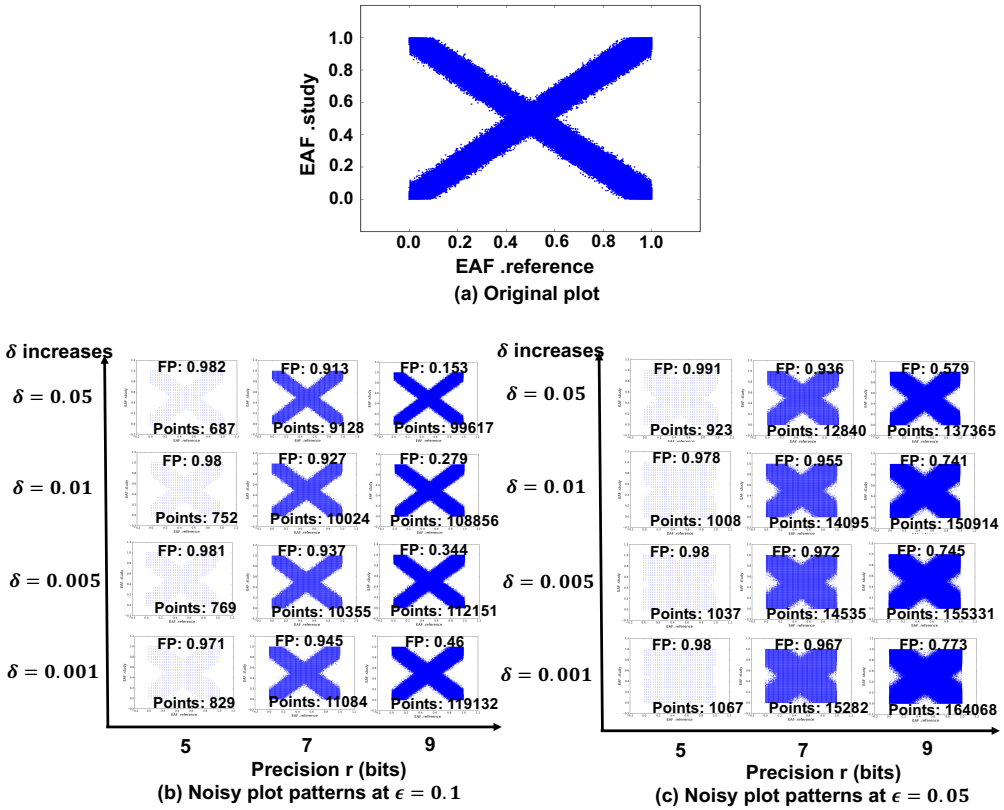

Figure 4.7: An example EAF plot and its noisy plot patterns for different levels of differential privacy and different precisions. (a) The original plot shows a study in which a fraction of the effect alleles was mis-specified. (b) Noisy plots when fixing $\epsilon = 0.1$ and changing δ and r. (c) Noisy plots when fixing $\epsilon = 0.05$ and changing δ and r. The number (e.g. 'Points: 687') at the bottom of each noisy plot is the number of EAF pairs output by SQC, namely, the number of points in the plot. The number (e.g. 'FP: 0.973') on top of each noisy plot indicates the false positive rate by applying the LLR test in Figure 4.5, when we pessimistically assume the strongest adversary that is able to link the EAF data back to their SNV identifiers. Plot patterns are well preserved when $\epsilon = 0.1$, while δ has a relatively modest effect and could be used to fine tune the pattern. At a low precision $r = 5$, the scatter plot is sparse and looks faded. As the precision becomes higher, the number of de-duplicated EAF pairs grows exponentially, and the false positive rate drops accordingly. An interesting phenomenon is that when δ increases (adding less noises), points deviate less from their original positions (Figure 4.6) and are more concentrated, hence the number of points decreases monotonically. However, the false positive rate is not monotonically decreasing when increasing δ, because fewer points lead to higher false positive rates (Figure 4.5). For example, at $\epsilon = 0.1$ and $r = 9$, when increasing δ from 0.005 to 0.01, its effect on the false positive rate is weaker than the effect of decreased number of points, hence the false positive rate becomes higher. The SQC framework can provide a reasonable trade-off by setting $\epsilon = 0.1, r = 7$, and then fine-tuning δ.

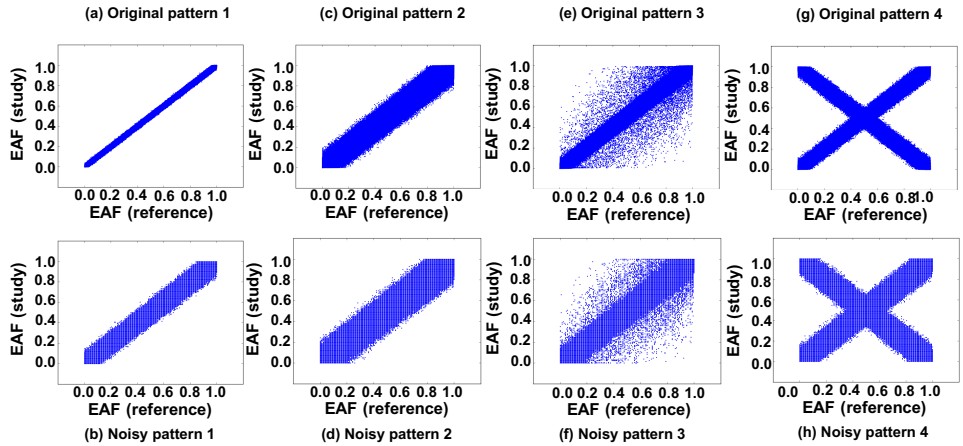

Figure 4.8: Original and noisy EAF plots. For each plot, the x-axis is the reference allele frequency of European ancestry, and the y-axis is the study allele frequency. Plots (a) and (c) show two studies with a relatively consistent European ancestry, while study (c) contains participants with a slightly different ancestry to the reference and has a thicker band. Plot (e) shows a study involving participants of non-European ancestry resulting in substantial deviation from the reference. Plot (g) shows a study with quality issues, e.g., a fraction of the effect alleles was mis-specified. And plots (b), (d), (f), and (h) are the corresponding noisy plots from SQC. A potential researcher can draw the same conclusion about the quality from these noisy plots as from the original plots.

on 64-bit numbers, which takes place in Protocol 2, but less than oblivious sorting on 96-bit numbers, which takes places in Protocol 3.

Small-scale experiments. To show the efficiency of each protocol on a single machine, we perform experiments with real data on small scales, varying from input of 1000 SNVs to 10 000 SNVs. In Figure 4.9-a, the secure P-Z protocol dominates the running time over the other two protocols, because of the secure division operations on 32-bit numbers and oblivious sorting on 96-bit numbers (64 bits for P-values and 32 bits for Z-statistics). Nevertheless, with a sequential implementation, these secure computation protocols seem impractical to run on a large-scale dataset that contains one million SNVs.

Parallel SQC. Due to the heavy overhead of secure computation and the non-trivial complexity of running the whole SQC protocols, it would be impractical to deploy the solution on a single machine with a sequential implementation. Building upon the ObliVM-GC backend, Nayak et al. propose a parallel secure-computation framework (GraphSC) that parallelizes graph-based algorithms and scales well in a cluster environment. We reconstruct SQC with the primitives provided by GraphSC, and deploy the system on a cluster of machines. The major benefit of using this paradigm is that oblivious sorting, which is the dominant overhead, can run in parallel. For our system configuration, we test the protocols on a cluster of 16 nodes, each equipped with Intel Xeon CPU E5-2680 v3 processors clocked at 2.5 GHz. Each machine consists of 8 cores and 32 GB of RAM. Half of the machines simulate cloud A, and the other half simulate cloud B. The bandwidth between machines is 1 Gbps.

In Figure 4.9-b, we show the performance of running the three SQC protocols on one

Procedure	Bits	Time	AND gates
Oblivious sorting	32	3.11s	1726976
	64	5.94s	3453952
	96	8.83s	5180928
Differential privacy	32	0.813s	192000
	64	1.472s	384000
	96	2.126s	576000
Precision reduction	32	3.57s	2015000
	64	6.7s	4127000
	96	9.84s	6239000
De-duplication (without sorting steps)	32	0.26s	94905
	64	0.4s	190809
	96	0.57s	286713
Secure division	32	5.46s	3230000
	64	20.09s	12606000
	96	46.19s	28126000

Table 4.1: Benchmark results of the computation steps, conditioning on 1000 SNVs. Oblivious sorting has a complexity of $O(n \log^2 n)$, whereas all other procedures have a linear blowup of complexity as the number of SNVs increase.

million SNVs. In total, executing the three quality control tasks takes about one hour, which we deem to be an acceptable cost, considering the days or even months of data access authorization procedures for biomedical studies. SQC safeguards the studies and minimizes the privacy concerns, producing a minimal interface where researchers retrieve sanitized and useful results in practical running time through parallel computation.

4.5 Discussion

Even though the data are not revealed as long as cloud A and cloud B do not collude, X_1 and X_2 should still be protected, considering that potential data breaches can occur on both clouds. The protection can be enforced on three levels, including hard disks, memory and cache. For example, on the level of the hard disk, HIPAA compliant cloud services could be one solution that normally encrypts the storage on disk. In this way, even if attackers steal the data on a disk from both clouds, they are not able to recover the original information. On the level of memory, the use of trusted hardware (e.g., Intel Software Guard eXtensions (SGX)) [22] has become an increasingly popular and powerful approach in recent years. This approach encrypts the memory so that attackers cannot steal the information, even by compromising the memory. There are more sophisticated attacks that occur, however, on the CPU cache level [41, 40]. There are different countermeasures proposed to thwart these attacks, but few of them are effective enough to be widely deployed in practice. The aforementioned cloud-protection methods are orthogonal to this work and could be added as components to our system.

For the effectiveness of hiding variants, we should emphasize that it is difficult to argue about the privacy guarantee of Procedure 1 alone, either theoretically or experimentally. Indeed, if the adversary has public reference statistics, it might be able to roughly map the identities by sorting the public statistics and comparing them with the sorted study statistics, but the precision of the mapping is highly data-dependent.

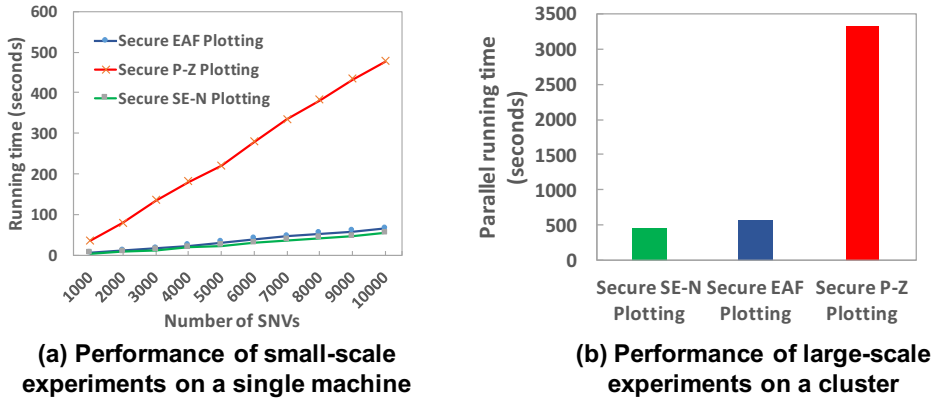

Figure 4.9: Runtime performance of three SQC protocols. (a) Running time of three protocols on small real datasets. It is measured on a single Intel Core i7 processor clocked at 3.1 GHz and 16 GB of RAM. We construct small datasets, varying from 1000 SNVs to 10000 SNVs, from the real data that contains 3 million SNVs. Secure P-Z plotting takes more time than the other two protocols because of secure division and oblivious sorting on numbers of 96 bits. This sequential implementation, especially due to sequential oblivious sorting, is not efficient enough to run on large datasets. (b) Parallel performance on two private clouds. Each machine has 8 cores clocked at 2.5 GHz and 32 GB of RAM, and each cloud has 8 machines (64 cores in total). Any two cores inside a cloud can communicate with each other, whereas a core of Cloud A can communicate with only one core of Cloud B in a channel with bandwidth of 1 Gbps, representing secure two-party computation as defined in Definition 1. The performance of the three SQC protocols is measured by processing one million SNVs. The most expensive secure P-Z protocol takes less than one hour.

Note that if the statistics under consideration describe pairwise SNV correlation (linkage disequilibrium), reducing precision might not be a sufficient measure to defend against some categories of attacks [12]. Indeed, if the victim is in the case group, such pairwise statistics from a few SNVs might contribute a substantial amount of information to attackers because it is less common to find a combination of two alleles than to find either one of them. Although these pairwise correlations are not seen in our data, researchers should take more precautions about revealing such correlation statistics than statistics of independent SNVs.

In parallel to SQC based on secure multiparty computation and differential privacy, homomorphic encryption is also widely used for untrusted cloud computing. Such an encryption enables an untrusted cloud to perform computation on randomized ciphertext, which is projected into certain computation on the corresponding plaintext. This is a highly desirable solution for our purpose of utilizing the cloud computing power without revealing the sensitive plaintext data, although existing efficient schemes are still constrained by the limited number of possible operations [167]. In the privacy-preserving genomic and medical studies, researchers have also proposed various systems based on homomorphic encryption [17, 28, 32, 31, 30].

4.6 Summary

Overall, SQC addresses the pressing issues of quality controlling privacy-preserving aggregate data sharing. By using a set of sanitization processes and advanced cryptographic tools, SQC guarantees that users of the framework have access to only minimal but sufficient output information that is unlikely to be useful for inference attacks. In particular, SQC automates the quality control phase in GWAS meta-analysis in a privacy-preserving manner. By projecting the quality-control protocols in a secure computation framework, SQC offers an effective balance between the needs of researchers for GWAS meta-analysis and the needs of data owners to respect the genetic privacy of research participants. Moreover, running SQC does not incur any utility loss for subsequent meta analyses. Although the strong security and privacy guarantees of SQC comes with intensive computation, we demonstrate that it is practical to perform the task using a recently proposed parallel secure computation framework. Considering the months-long discussions required for data access agreement, the SQC framework, once established, is promising for automating both the data sharing and protection, hence paving the way for large-scale medical research.

Chapter 5

Conclusion

In this thesis, we have investigated the potential threats of cloud-based genomic computing systems and have proposed various countermeasures by taking into account the functionality requirements. We have shown how to choose appropriate privacy-enhancing techniques in order to design the privacy-conscious counterpart of existing cloud-based genomic computing systems. Different techniques have different advantages and limitations, which system designers must carefully consider in order to avoid pitfalls (e.g., security vulnerabilities, inefficiency, low accuracy).

In Chapter 2, we have presented a robust cloud storage system of genomic data for end users who can employ password-based encryption to protect their data from brute-force attacks. We have proposed GenoGuard to defend against data breaches that involve a computationally unbounded adversary who can brute-force passwords or even cryptographic keys. With various data breaches showing that users tend to choose weak passwords, GenoGuard is particularly useful in the case of the password-based encryption that is frequently used by direct-to-consumer services. GenoGuard is built on honey encryption, where plaintext messages need to be transformed to a different space with uniform distribution on elements. We designed a novel distribution-transformation encoder for honey encryption. This encoder employs a hidden Markov chain on genome sequence, and our chi-square test shows that the model fits well on the distribution of genotypes. We have also presented a thorough security proof on the resulting honey encryption scheme that uses our new distribution-transformation encoder. To ensure a security loss smaller than 2^{-200}, GenoGuard needs to expand the storage by only a factor slightly larger than 2; in exchange for this, users gain the security property against brute-force attacks on the encrypted data. We have also shown the high probability of being able to break the conventional password-based encryption scheme if the password set is limited (e.g., 1000 passwords), whereas GenoGuard remains robust in the same scenario. In addition, we have proposed and analyzed techniques for preventing an adversary from exploiting phenotype information (physical traits of victims) in order to decrypt genomes. For example, with the "red" hair information, an adversary's advantage can increase from 0.0379 to 0.0642. The countermeasure to this problem is to incorporate the side information during the encoding phase, which we have demonstrated for the case of ancestry information. Our Python implementation requires around 1 minute to encrypt 22 chromosomes on a cluster of 22 nodes.

In Chapter 3, we have introduced a privacy-preserving solution, SECRAM: it offers efficient random retrieval on large encrypted genomic data. Researchers or doctors are sometimes interested in retrieving the highly redundant raw genomic data, in addition to genetic variants, in order to discover some other useful information or sequencing errors. With 3 billion nucleotides being replicated and stored multiple times in a file (sometimes even randomly located), the data of one person can go up to several hundred gigabytes, posing two major technical challenges: storage costs and data processing efficiency. In this regard, we have presented a privacy-preserving solution for the secure storage and efficient retrieval of compressed aligned genomic data. We have shown the possibility of storing read-based genomic data in a transposed manner, and of still maintaining all information with lossless compression. Compared with BAM (the current standard), SECRAM provides strong protection and saves 18% of storage on average, when the coverage is higher than 10×. For example, BAM uses 472 MB to store human chromosome 11 with 3× coverage, whereas SECRAM only uses 407 MB. SECRAM also offers efficient data retrieval that requires less than 0.25 seconds to obtain 1 million genomic positions with a 3× coverage. We maintain both efficient compression and downstream data processing in SECRAM, and we provide unprecedented levels of security in genomic data storage. SECRAM thus offers a space-saving, privacy-preserving, and efficient solution for the storage of clinical genomic data.

Furthermore, SECRAM is a result of the collaboration project with the Swiss company Sophia Genetics that specialises in genomic analysis and data-driven medicine. The biggest challenge and also most important lesson we learnt from this collaboration is: the bioinformatics community has well-established storage formats and query interfaces, hence any newly designed framework (with security and privacy features) has to comply with those for smooth integration with real applications. Our design is based on a thorough evaluation of real deployment scenarios and requirements, which renders SECRAM a highly compliant product for existing cloud systems. We have also discussed the potential vulnerability of using order-preserving encryption. Although SECRAM offers a smooth transition from current practice to a more privacy-conscious system, we envision more secure cryptographic primitives to be integrated, such as searchable encryption.

In Chapter 4, we have described SQC in detail for securing genomic-data quality control during genome-wide association meta-analysis. To some extent, fear-induced regulations thwart the development of genomic analysis, especially after the revelation of a number of powerful statistical inference attacks on aggregated data. As an alternative to the time-consuming data-sharing logistics, SQC demonstrates another possibility that the data-analysis collaboration can be enabled in a secure manner by employing advanced cryptographic techniques, such as secure multi-party computation based on garbled circuit. Certainly, these techniques come at a high computational cost, but it is modest compared to existing practices of data sharing. We have shown that the expected patterns of data quality plots will have only a negligible change even after ensuring strong differential privacy, rendering SQC a robust scheme for privacy-preserving data sharing. The system makes use of parallel processing. To demonstrate the efficiency and scalability on commodity machines, we have implemented the solution in a meta-analysis pipeline with real data. As a result, we achieved a performance of one-hour execution for three quality control protocols for one million genetic variants, on a cluster of 128 cores. With rapid advances in the cryptographic community, we expect the performance of these advanced schemes to be much better in the near future, opening the possi-

bility that efficient secure data-sharing protocols could substitute certain fear-induced regulations.

In conclusion, we have devised various privacy-enhancing tools that data scientists can exploit when facing a privacy and security requirement, and we have shown several concrete privacy-conscious systems that employ these tools. As the sensitive nature of genomic data becomes increasingly more visible to attackers, genomic-data protection becomes more relevant to cloud computing. Nevertheless, despite its cutting-edge interdisciplinary research, privacy-conscious genomic computing systems are mostly based on many techniques available in the cryptography community. Communication between the genomic community and the cryptography community is of paramount importance. In absence of this, hardly any solution can claim itself secure, useful, or even meaningful. The state of the art in data hacking continues to advance, demanding ongoing efforts to protect against the latest vulnerabilities. After completing this thesis, we can breathe a sigh of relief because of the extensive powerful privacy-enhancing tools available, if only on our academic shelves. But at the same time, we must persevere in further research into cloud computing for genomic data due to the subtlety of privacy-enhancing designs, such as efficiency, security, privacy level, accuracy, storage cost, and compatibility. Meanwhile, we should educate the general public about the risks so that they know to demand for better security and privacy in genomic applications. Surely, the genomic-privacy community will remain busy for years to come, dealing with all these problems and those as yet unimagined.

Bibliography

[1] https://cloud.google.com/genomics/, [Online; accessed 13-November-2014]. [cited at p. 1]

[2] https://www.ibm.com/watson/health/oncology-and-genomics/genomics/. [cited at p. 1]

[3] https://enterprise.microsoft.com/en-us/industries/health/genomics/. [cited at p. 1]

[4] https://www.apple.com/newsroom/2016/03/21Apple-Announces-Advancements-to-ResearchKit/. [cited at p. 1]

[5] https://aws.amazon.com/health/genomics/. [cited at p. 1]

[6] https://www.nytimes.com/2016/12/14/technology/yahoo-hack.html?action=Click&contentCollection=BreakingNews&contentID=64651831&pgtype=Homepage&_r=0. [cited at p. 1]

[7] http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/. [cited at p. 1]

[8] http://www.nytimes.com/2013/12/29/business/reading-your-palm-for-securitys-sake.html. [cited at p. 1]

[9] https://www.forbes.com/sites/matthewherper/2015/01/06/surprise-with-60-million-genentech-deal-23andme-has-a-business-plan/. [cited at p. 2]

[10] http://www.nature.com/news/privacy-protections-the-genome-hacker-1.12940. [cited at p. 2]

[11] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V. Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig, "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. [cited at p. 2, 3, 62]

[12] R. Wang, Y. F. Li, X. Wang, H. Tang, and X. Zhou, "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*, 2009, pp. 534–544. [cited at p. 2, 3, 62, 82]

[13] S. E. Fienberg, A. B. Slavkovic, and C. Uhler, "Privacy preserving GWAS data sharing," in *IEEE 11th International Conference on Data Mining Workshops (ICDMW)*, 2011, pp. 628–635. [cited at p. 2, 3, 64, 68]

[14] 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*, 2013, pp. 1079–1087. [cited at p. 2, 3, 68]

[15] F. Yu, S. E. Fienberg, A. B. Slavković, and C. Uhler, "Scalable privacy-preserving data sharing methodology for genome-wide association studies," *Journal of biomedical informatics*, 2014. [cited at p. 2, 3]

[16] 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*. ACM, 2015, pp. 1286–1297. [cited at p. 2, 3, 68]

[17] E. Ayday, J. L. Raisaro, J.-P. Hubaux, and J. Rougemont, "Protecting and evaluating genomic privacy in medical tests and personalized medicine," in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, 2013, pp. 95–106. [cited at p. 2, 3, 82]

[18] 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. [cited at p. 2, 3]

[19] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik, "Countering GATTACA: Efficient and secure testing of fully-sequenced human genomes," in *Proceedings of the 18th ACM conference on Computer and communications security*, 2011, pp. 691–702. [cited at p. 2, 3]

[20] 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. [cited at p. 2, 4]

[21] F. Chen, S. Wang, X. Jiang, S. Ding, Y. Lu, J. Kim, S. C. Sahinalp, C. Shimizu, J. C. Burns, V. J. Wright *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. [cited at p. 2, 4]

[22] F. Chen, C. Wang, W. Dai, X. Jiang, N. Mohammed, M. M. Al Aziz, M. N. Sadat, C. Sahinalp, K. Lauter, and S. Wang, "PRESAGE: Privacy-preserving genetic testing via software guard extension," *BMC medical genomics*, vol. 10, no. 2, p. 48, 2017. [cited at p. 2, 4, 81]

[23] M. Naveed, E. Ayday, E. W. Clayton, J. Fellay, C. A. Gunter, J.-P. Hubaux, B. A. Malin, and X. Wang, "Privacy and security in the genomic era," *arXiv preprint arXiv:1405.1891*, 2014. [cited at p. 2]

[24] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *USENIX Security*, 2014. [cited at p. 3, 68]

[25] Y. Erlich and A. Narayanan, "Routes for breaching and protecting genetic privacy," *Nature Reviews Genetics*, vol. 15, no. 6, pp. 409–421, 2014. [cited at p. 2, 3]

[26] S. Jha, L. Kruger, and V. Shmatikov, "Towards practical privacy for genomic computation," in *IEEE Symposium on Security and Privacy*, 2008, pp. 216–230. [cited at p. 3]

[27] R. Wang, X. Wang, Z. Li, H. Tang, M. K. Reiter, and Z. Dong, "Privacy-preserving genomic computation through program specialization," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 338–347. [cited at p. 3]

[28] E. Ayday, J. L. Raisaro, P. J. Mclaren, J. Fellay, and J.-p. Hubaux, "Privacy-preserving computation of disease risk by using genomic, clinical, and environmental data," in *Proceedings of USENIX Security Workshop on Health Information Technologies (HealthTech)*, 2013. [cited at p. 3, 82]

[29] W. Xie, M. Kantarcioglu, W. S. Bush, D. Crawford, J. C. Denny, R. Heatherly, and B. A. Malin, "SecureMA: protecting participant privacy in genetic association meta-analysis," *Bioinformatics*, vol. 30, no. 23, pp. 3334–3341, Dec. 2014. [cited at p. 3, 62, 66]

[30] M. Kim and K. Lauter, "Private genome analysis through homomorphic encryption," *BMC Medical Informatics and Decision Making*, vol. 15, no. 5, p. S3, 2015. [cited at p. 3, 82]

[31] K. Shimizu, K. Nuida, and G. Rtsch, "Efficient privacy-preserving string search and an application in genomics," *Bioinformatics*, vol. 32, no. 11, pp. 1652–1661, Jun. 2016. [cited at p. 3, 82]

[32] S. Wang, Y. Zhang, W. Dai, K. Lauter, M. Kim, Y. Tang, H. Xiong, and X. Jiang, "HEALER: Homomorphic computation of ExAct Logistic rEgRession for secure rare disease variants analysis in GWAS," *Bioinformatics*, vol. 32, no. 2, pp. 211–218, Jan. 2016. [cited at p. 3, 82]

[33] D. Froelicher, P. Egger, J. S. Sousa, J. L. Raisaro, Z. Huang, C. Mouchet, B. Ford, and J.-P. Hubaux, "UnLynx: A decentralized system for privacy-conscious data sharing," in *Proceedings on Privacy Enhancing Technologies*, vol. 4, 2017, pp. 152–170. [cited at p. 3]

[34] J. R. Troncoso-Pastoriza, A. Pedrouzo-Ulloa, and F. Pérez-González, "Secure genomic susceptibility testing based on lattice encryption," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 2067–2071. [cited at p. 3]

[35] J. S. Sousa, C. Lefebvre, Z. Huang, J. L. Raisaro, C. Aguilar-Melchor, M.-O. Killijian, and J.-P. Hubaux, "Efficient and secure outsourcing of genomic data storage," *BMC Medical Genomics*, vol. 10, no. Suppl 2, Jul. 2017. [cited at p. 3]

[36] K. A. Jagadeesh, D. J. Wu, J. A. Birgmeier, D. Boneh, and G. Bejerano, "Deriving genomic diagnoses without revealing patient genomes," *Science*, vol. 357, no. 6352, pp. 692–695, 2017. [cited at p. 3]

[37] F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, and M. Russinovich, "VC3: Trustworthy Data Analytics in the Cloud Using SGX," in *2015 IEEE Symposium on Security and Privacy (SP)*, May 2015, pp. 38–54. [cited at p. 4]

[38] T. T. A. Dinh, P. Saxena, E.-C. Chang, B. C. Ooi, and C. Zhang, "M2r: Enabling Stronger Privacy in Mapreduce Computation," in *Proceedings of the 24th USENIX Conference on Security Symposium*, 2015, pp. 447–462. [cited at p. 4]

[39] O. Ohrimenko, M. Costa, C. Fournet, C. Gkantsidis, M. Kohlweiss, and D. Sharma, "Observing and Preventing Leakage in MapReduce," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1570–1581. [cited at p. 4]

[40] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-Tenant Side-Channel Attacks in PaaS Clouds," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14, 2014, pp. 990–1003. [cited at p. 4, 81]

[41] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee, "Last-Level Cache Side-Channel Attacks are Practical," in *2015 IEEE Symposium on Security and Privacy*, May 2015, pp. 605–622. [cited at p. 4, 81]

[42] Y. Xu, W. Cui, and M. Peinado, "Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems," in *2015 IEEE Symposium on Security and Privacy*, May 2015, pp. 640–656. [cited at p. 4]

[43] A. Moghimi, G. Irazoqui, and T. Eisenbarth, "Cachezoom: How sgx amplifies the power of cache attacks," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 69–90. [cited at p. 4]

[44] https://www.wired.com/story/meltdown-spectre-bug-collision-intel-chip-flaw-discovery/. [cited at p. 4]

[45] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown," *ArXiv e-prints*, Jan. 2018. [cited at p. 4]

[46] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," *ArXiv e-prints*, Jan. 2018. [cited at p. 4]

[47] Z. Huang, E. Ayday, J. Fellay, J.-P. Hubaux, and A. Juels, "GenoGuard: Protecting genomic data against brute-force attacks," in *36th IEEE Symposium on Security and Privacy (SP)*, 2015, pp. 447–462. [cited at p. 7]

[48] Z. Huang, E. Ayday, H. Lin, R. S. Aiyar, A. Molyneaux, Z. Xu, J. Fellay, L. M. Steinmetz, and J.-P. Hubaux, "A privacy-preserving solution for compressed storage and selective retrieval of genomic data," *Genome research*, vol. 26, no. 12, pp. 1687–1696, 2016. [cited at p. 7]

[49] Z. Huang, H. Lin, J. Fellay, Z. Kutalik, and J.-P. Hubaux, "SQC: secure quality control for meta-analysis of genome-wide association studies," *Bioinformatics*, vol. 33, no. 15, pp. 2273–2280, 2017. [cited at p. 7]

[50] https://www.cleardata.com/wp-content/uploads/2016/12/2016-HIMSS-Analytics-Cloud-Study.pdf, [Online; accessed 2-October-2017]. [cited at p. 9]

[51] https://www.ispartnersllc.com/blog/most-trusted-hipaa-compliant-cloud-storage-services/, [Online; accessed 2-October-2017]. [cited at p. 9]

[52] https://www.nbcnews.com/technology/your-cloud-drive-really-private-not-according-fine-print-1C8881731, [Online; accessed 2-October-2017]. [cited at p. 9]

[53] https://lifehacker.com/the-best-cloud-storage-services-that-protect-your-priva-729639300, [Online; accessed 2-October-2017]. [cited at p. 9]

[54] https://www.forbes.com/sites/kellyclay/2012/07/19/is-microsoft-spying-on-skydrive-users/, [Online; accessed 2-October-2017]. [cited at p. 9]

[55] https://spideroak.com/. [cited at p. 9]

[56] https://www.comparitech.com/blog/cloud-online-backup/6-apps-to-encrypt-your-files-before-uploading-to-the-cloud/, [Online; accessed 2-October-2017]. [cited at p. 9]

[57] https://www.ibm.com/support/knowledgecenter/en/SSEPGG_9.5.0/com.ibm.db2.luw.admin.sec.doc/doc/c0005815.html, [Online; accessed 2-October-2017]. [cited at p. 9]

[58] https://support.office.com/en-us/article/Encrypt-a-database-by-using-a-database-password-fe1cc5fe-f9a5-4784-b090-fdb2673457ab, [Online; accessed 2-October-2017]. [cited at p. 9]

[59] 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 ACM SIGSAC conference on Computer and communications security*, 2013, pp. 1141–1152. [cited at p. 10]

[60] D. Florencio and C. Herley, "A large-scale study of web password habits," in *Proceedings of the 16th International Conference on World Wide Web*, ser. WWW '07. New York, NY, USA: ACM, 2007, pp. 657–666. [Online]. Available: http://doi.acm.org/10.1145/1242572.1242661 [cited at p. 10, 11]

[61] A. Juels and T. Ristenpart, "Honey encryption: Security beyond the brute-force bound," in *Advances in Cryptology–EUROCRYPT*, 2014, pp. 293–310. [cited at p. 4, 10, 11, 12, 13, 26, 27]

[62] J. M. VanLiere and N. A. Rosenberg, "Mathematical properties of the r^2 measure of linkage disequilibrium," *Theoretical population biology*, vol. 74, no. 1, pp. 130–137, 2008. [cited at p. 12]

[63] M. Benantar, *Access control systems: Security, identity management and trust models*. Springer, 2006. [cited at p. 14]

[64] B. Kaliski, *PKCS# 5: Password-based cryptography specification version 2.0*, RSA Laboratories, September, 2000. [cited at p. 18, 29, 35]

[65] M. S. McPeek and A. Strahs, "Assessment of linkage disequilibrium by the decay of haplotype sharing, with application to fine-scale genetic mapping," *The American Journal of Human Genetics*, vol. 65, pp. 858–875, 1999. [cited at p. 20]

[66] S. L. Salzberg, A. L. Delcher, S. Kasif, and O. White, "Microbial gene identification using interpolated Markov models," *Nucleic acids research*, vol. 26, pp. 544–548, 1998. [cited at p. 20]

[67] N. Li and M. Stephens, "Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data," *Genetics*, vol. 165, pp. 2213–2233, 2003. [cited at p. 20]

[68] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, 1989. [cited at p. 21]

[69] J. Marchini, B. Howie, S. Myers, G. McVean, and P. Donnelly, "A new multipoint method for genome-wide association studies by imputation of genotypes," *Nature genetics*, vol. 39, pp. 906–913, 2007. [cited at p. 21, 22]

[70] http://hapmap.ncbi.nlm.nih.gov/downloads/index.html.en, [Online; accessed 11-November-2014]. [cited at p. 22, 31, 35]

[71] R. B. DiAgostino and J. M. Massaro, "Goodness-of-fit tests," *Handbook of the Logistic Distribution*, p. 327, 2013. [cited at p. 23]

[72] J. Bonneau, "The science of guessing: Analyzing an anonymized corpus of 70 million passwords," in *IEEE Symposium on Security and Privacy*, 2012, pp. 538–552. [cited at p. 28]

[73] P. Claes, D. K. Liberton, K. Daniels, K. M. Rosana, E. E. Quillen, L. N. Pearson, B. McEvoy, M. Bauchet, A. A. Zaidi, W. Yao *et al.*, "Modeling 3D facial shape from DNA," *PLoS genetics*, March 20, 2014. [cited at p. 30]

[74] A. L. Price, N. J. Patterson, R. M. Plenge, M. E. Weinblatt, N. A. Shadick, and D. Reich, "Principal components analysis corrects for stratification in genome-wide association studies," *Nature genetics*, vol. 38, no. 8, pp. 904–909, 2006. [cited at p. 31]

[75] J. N. Sampson, K. K. Kidd, J. R. Kidd, and H. Zhao, "Selecting SNPs to identify ancestry," *Annals of human genetics*, vol. 75, no. 4, pp. 539–553, 2011. [cited at p. 31]

[76] D. Malone and K. Maher, "Investigating the distribution of password choices," in *Proceedings of the 21st international conference on World Wide Web*. ACM, 2012, pp. 301–310. [cited at p. 32]

[77] D. Wang, G. Jian, H. Cheng, Q. Gu, C. Zhu, and P. Wang, "Zipfs law in passwords," Cryptology ePrint Archive, Report 2014/631, Tech. Rep., 2014. [cited at p. 32, 33]

[78] S. Walsh, F. Liu, A. Wollstein, L. Kovatsi, A. Ralf, A. Kosiniak-Kamysz, W. Branicki, and M. Kayser, "The HIrisPlex system for simultaneous prediction of hair and eye colour from DNA," *Forensic Science International: Genetics*, vol. 7, no. 1, pp. 98–115, 2013. [cited at p. 33]

[79] F. Haist, A. P. Shimamura, and L. R. Squire, "On the relationship between recall and recognition memory," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 18, no. 4, p. 691, 1992. [cited at p. 37]

[80] R. Dhamija and J. D. Tygar, "The battle against phishing: Dynamic security skins," in *Proceedings of Symposium on Usable Privacy and Security*, 2005, pp. 77–88. [cited at p. 37]

[81] R. Cappelli, A. Erol, D. Maio, and D. Maltoni, "Synthetic fingerprint-image generation," in *Proceedings of the 15th International Conference on Pattern Recognition*, vol. 3, 2000, pp. 471–474. [cited at p. 37]

[82] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge University Press, 2010. [cited at p. 4]

[83] L. S. Clair, L. Johansen, W. Enck, M. Pirretti, P. Traynor, P. McDaniel, and T. Jaeger, "Password exhaustion: Predicting the end of password usefulness," in *Information Systems Security*, 2006, pp. 37–55. [cited at p. 4]

[84] M. L. Mazurek, S. Komanduri, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, P. G. Kelley, R. Shay, and B. Ur, "Measuring password guessability for an entire university," in *Proceedings of the ACM SIGSAC conference on Computer & communications security*, 2013, pp. 173–186. [cited at p. 4]

[85] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, "The tangled web of password reuse," in *Proceedings of Network and Distributed System Security Symposium*, 2014. [cited at p. 4]

[86] L. Spitzner, *Honeypots: Tracking hackers*. Addison-Wesley Reading, 2003. [cited at p. 4]

[87] C. Kreibich and J. Crowcroft, "Honeycomb: Creating intrusion detection signatures using honeypots," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 51–56, 2004. [cited at p. 4]

[88] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levine, and H. Owen, "Honeystat: Local worm detection using honeypots," in *Recent Advances in Intrusion Detection*, 2004, pp. 39–58. [cited at p. 4]

[89] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. P. Markatos, and A. D. Keromytis, "Detecting targeted attacks using shadow honeypots." in *Usenix Security*, 2005. [cited at p. 4]

[90] J. Levine, R. LaBella, H. Owen, D. Contis, and B. Culver, "The use of honeynets to detect exploited systems across large enterprise networks," in *IEEE Systems, Man and Cybernetics Society Information Assurance Workshop*, 2003, pp. 92–99. [cited at p. 4]

[91] H. Bojinov, E. Bursztein, X. Boyen, and D. Boneh, "Kamouflage: Loss-resistant password management," in *ESORICS*, 2010. [cited at p. 4]

[92] A. Juels and R. L. Rivest, "Honeywords: Making password-cracking detectable," in *Proceedings of the ACM SIGSAC conference on Computer & communications security*, 2013, pp. 145–160. [cited at p. 4]

[93] R. Chatterjee, J. Bonneau, A. Juels, and T. Ristenpart, "Cracking-resistant password vaults using natural language encoders," in *36th IEEE Symposium on Security and Privacy (SP)*, 2015, pp. 481–498. [cited at p. 4]

[94] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner, "Dynamic searchable encryption in very-large databases: Data structures and implementation." in *NDSS*, vol. 14, 2014, pp. 23–26. [cited at p. 39]

[95] E. Pennisi, "Will Computers Crash Genomics?" *Science*, vol. 331, no. 6018, pp. 666–668, Feb. 2011. [cited at p. 39]

[96] http://www.ebi.ac.uk/ena/software/cram-toolkit. [cited at p. 40, 45, 58]

[97] http://www.genome.gov/sequencingcosts/. [cited at p. 40]

[98] Z. D. Stephens, S. Y. Lee, F. Faghri, R. H. Campbell, C. Zhai, M. J. Efron, R. Iyer, M. C. Schatz, S. Sinha, and G. E. Robinson, "Big Data: Astronomical or Genomical?" *PLoS Biol*, vol. 13, no. 7, p. e1002195, Jul. 2015. [cited at p. 40]

[99] https://aws.amazon.com/s3/pricing/. [cited at p. 40]

[100] Z. Zhu, Y. Zhang, Z. Ji, S. He, and X. Yang, "High-throughput DNA sequence data compression," *Briefings in Bioinformatics*, vol. 16, no. 1, pp. 1–15, Jan. 2015. [cited at p. 40, 41]

[101] S. Grumbach and F. Tahi, "Compression of DNA sequences," in *Data Compression Conference, 1993. DCC '93.*, 1993, pp. 340–350. [cited at p. 40]

[102] X. Chen, S. Kwong, and M. Li, "A Compression Algorithm for DNA Sequences and Its Applications in Genome Comparison," in *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*. New York, NY, USA: ACM, 2000. [cited at p. 40]

[103] X. Chen, M. Li, B. Ma, and J. Tromp, "DNACompress: fast and effective DNA sequence compression," *Bioinformatics*, vol. 18, no. 12, pp. 1696–1698, 2002. [cited at p. 40]

[104] S. Christley, Y. Lu, C. Li, and X. Xie, "Human genomes as email attachments," *Bioinformatics*, vol. 25, no. 2, pp. 274–275, Jan. 2009. [cited at p. 40]

[105] https://www.gzip.org. [cited at p. 40]

[106] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and . G. P. D. P. Subgroup, "The Sequence Alignment/Map format and SAMtools," *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, Aug. 2009. [cited at p. 40, 46]

[107] D. C. Jones, W. L. Ruzzo, X. Peng, and M. G. Katze, "Compression of next-generation sequencing reads aided by highly efficient de novo assembly," *Nucleic Acids Research*, p. gks754, Aug. 2012. [cited at p. 40]

[108] J. K. Bonfield and M. V. Mahoney, "Compression of FASTQ and SAM Format Sequencing Data," *PLoS ONE*, vol. 8, no. 3, p. e59190, Mar. 2013. [cited at p. 40]

[109] P. Li, X. Jiang, S. Wang, J. Kim, H. Xiong, and L. Ohno-Machado, "HUGO: Hierarchical mUlti-reference Genome cOmpression for aligned reads," *Journal of the American Medical Informatics Association*, vol. 21, no. 2, pp. 363–373, Mar. 2014. [cited at p. 40]

[110] M. H.-Y. Fritz, R. Leinonen, G. Cochrane, and E. Birney, "Efficient storage of high throughput DNA sequencing data using reference-based compression," *Genome Research*, vol. 21, no. 5, pp. 734–740, May 2011. [cited at p. 40]

[111] M. Massie, F. Nothaft, C. Hartl, C. Kozanitis, A. Schumacher, A. D. Joseph, and D. A. Patterson, "Adam: Genomics formats and processing patterns for cloud scale computing," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2013-207*, 2013. [cited at p. 41]

[112] Y. W. Yu, D. Yorukoglu, J. Peng, and B. Berger, "Quality score compression improves genotyping accuracy," *Nature Biotechnology*, vol. 33, no. 3, pp. 240–243, Mar. 2015. [cited at p. 41, 58]

[113] S. D. Kahn, "On the Future of Genomic Data," *Science*, vol. 331, no. 6018, pp. 728–729, Feb. 2011. [cited at p. 2]

[114] E. L. van Dijk, H. Auger, Y. Jaszczyszyn, and C. Thermes, "Ten years of next-generation sequencing technology," *Trends in Genetics*, vol. 30, no. 9, pp. 418–426, Sep. 2014. [cited at p. 2]

[115] E. Ayday, J. L. Raisaro, U. Hengartner, A. Molyneaux, and J.-P. Hubaux, "Privacy-Preserving Processing of Raw Genomic Data," in *8th International Workshop on Data Privacy Management and Autonomous Spontaneous Security*, New York, NY, USA, 2013, no. 8247, pp. 133–147. [cited at p. 2, 41, 42, 47, 56, 57]

[116] G. Onsongo, J. Erdmann, M. D. Spears, J. Chilton, K. B. Beckman, A. Hauge, S. Yohe, M. Schomaker, M. Bower, K. A. T. Silverstein, and B. Thyagarajan, "Implementation of Cloud based Next Generation Sequencing data analysis in a clinical laboratory," *BMC Research Notes*, vol. 7, no. 1, p. 314, May 2014. [cited at p. 2]

[117] J. G. Reid, A. Carroll, N. Veeraraghavan, M. Dahdouli, A. Sundquist, A. English, M. Bainbridge, S. White, W. Salerno, C. Buhay, F. Yu, D. Muzny, R. Daly, G. Duyk, R. A. Gibbs, and E. Boerwinkle, "Launching genomics into the cloud: deployment of Mercury, a next generation sequence analysis pipeline," *BMC Bioinformatics*, vol. 15, no. 1, p. 30, Jan. 2014. [cited at p. 2]

[118] Z. Rilak, S. Wernicke, and I. Bogicevic, "Keeping Genomic Data Safe on the Cloud," *Journal of Biomolecular Techniques : JBT*, vol. 25, no. Suppl, p. S5, May 2014. [cited at p. 2]

[119] https://samtools.github.io/hts-specs/SAMv1.pdf. [cited at p. 42]

[120] A. Boldyreva, N. Chenette, and A. ONeill, "Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions," in *Advances in Cryptology CRYPTO 2011*, Aug. 2011, no. 6841, pp. 578–595. [cited at p. 46, 49, 57]

[121] H. Lipmaa, D. Wagner, and P. Rogaway, *Comments to NIST concerning AES modes of operation: CTR-mode encryption*, 2000. [cited at p. 47]

[122] https://samtools.github.io/htsjdk/. [cited at p. 51]

[123] https://www.bouncycastle.org/. [cited at p. 51]

[124] http://www.sophiagenetics.com/home.html. [cited at p. 54]

[125] V. Kolesnikov and A. Shikfa, "On The Limits of Privacy Provided by Order-Preserving Encryption," *Bell Labs Technical Journal*, vol. 17, no. 3, pp. 135–146, Dec. 2012. [cited at p. 57]

[126] M. Naveed, S. Kamara, and C. V. Wright, "Inference Attacks on Property-Preserving Encrypted Databases," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15, New York, NY, USA, 2015, pp. 644–655. [cited at p. 57]

[127] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order Preserving Encryption for Numeric Data," in *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '04, New York, NY, USA, 2004, pp. 563–574. [cited at p. 57]

[128] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu, "Practical Order-Revealing Encryption with Limited Leakage," Cryptology ePrint Archive, Tech. Rep. 1125, 2015. [cited at p. 57]

[129] F. Kerschbaum, "Frequency-Hiding Order-Preserving Encryption," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15, New York, NY, USA, 2015, pp. 656–667. [cited at p. 57]

[130] D. S. Roche, D. Apon, S. G. Choi, and A. Yerukhimovich, "POPE: Partial Order Preserving Encoding," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16, New York, NY, USA, 2016, pp. 1131–1142. [cited at p. 57]

[131] http://www.pacb.com/products-and-services/pacbio-systems/sequel/. [cited at p. 58]

[132] T. W. Winkler, F. R. Day, D. C. Croteau-Chonka, A. R. Wood, A. E. Locke, R. Mägi, T. Ferreira, T. Fall, M. Graff, A. E. Justice *et al.*, "Quality control and conduct of genome-wide association meta-analyses," *Nature Protocols*, vol. 9, no. 5, pp. 1192–1212, 2014. [cited at p. 62, 71, 72]

[133] H. K. Im, E. R. Gamazon, D. L. Nicolae, and N. J. Cox, "On Sharing Quantitative Trait GWAS Results in an Era of Multiple-omics Data and the Limits of Genomic Privacy," *The American Journal of Human Genetics*, vol. 90, no. 4, pp. 591–598, Apr. 2012. [cited at p. 3, 62]

[134] K. B. Jacobs, M. Yeager, S. Wacholder, D. Craig, P. Kraft, D. J. Hunter, J. Paschal, T. A. Manolio, M. Tucker, R. N. Hoover *et al.*, "A new statistic and its power to infer membership in a genome-wide association study using genotype frequencies," *Nature genetics*, vol. 41, no. 11, pp. 1253–1257, 2009. [cited at p. 3, 62]

[135] Lumley T and Rice K, "Potential for revealing individual-level information in genome-wide association studies," *JAMA*, vol. 303, no. 7, pp. 659–660, Feb. 2010. [cited at p. 3, 62]

[136] S. Sankararaman, G. Obozinski, M. I. Jordan, and E. Halperin, "Genomic privacy and limits of individual detection in a pool," *Nature genetics*, vol. 41, no. 9, pp. 965–967, 2009. [cited at p. 3, 62, 74]

[137] P. M. Visscher and W. G. Hill, "The Limits of Individual Identification from Sample Allele Frequencies: Theory and Statistical Analysis," *PLOS Genet*, vol. 5, no. 10, p. e1000628, Oct. 2009. [cited at p. 3, 62]

[138] E. A. Zerhouni and E. G. Nabel, "Protecting aggregate genomic data," *Science*, vol. 322, no. 5898, p. 44a, 2008. [cited at p. 62]

[139] A. Yao, "How to generate and exchange secrets," in *Foundations of Computer Science, 1986., 27th Annual Symposium on.* IEEE, 1986, pp. 162–167. [cited at p. 63]

[140] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols," in *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, 2001, pp. 448–457. [cited at p. 64]

[141] V. Kolesnikov and T. Schneider, "Improved Garbled Circuit: Free XOR Gates and Applications," in *Automata, Languages and Programming*, Jul. 2008, pp. 486–498. [cited at p. 64]

[142] S. Zahur, M. Rosulek, and D. Evans, "Two Halves Make a Whole," in *Advances in Cryptology - EUROCRYPT 2015.* Springer, Berlin, Heidelberg, Apr. 2015, pp. 220–250. [cited at p. 64]

[143] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending Oblivious Transfers Efficiently," in *Advances in Cryptology - CRYPTO 2003*, Aug. 2003, pp. 145–161. [cited at p. 64]

[144] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More Efficient Oblivious Transfer and Extensions for Faster Secure Computation," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, 2013, pp. 535–548. [cited at p. 64]

[145] V. Kolesnikov and R. Kumaresan, "Improved OT Extension for Transferring Short Secrets," in *Advances in Cryptology CRYPTO 2013*, 2013, pp. 54–70. [cited at p. 64]

[146] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplaya Secure Two-party Computation System," in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, 2004, pp. 20–20. [cited at p. 64]

[147] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster Secure Two-party Computation Using Garbled Circuits," in *Proceedings of the 20th USENIX Conference on Security*, 2011, pp. 35–35. [cited at p. 64]

[148] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014. [cited at p. 64, 69]

[149] A. P. Singh, S. Zafer, and I. Peer, "Metaseq: privacy preserving meta-analysis of sequencing-based association studies," in *Pacific Symposium on Biocomputing*, vol. 356, 2013, p. 367. [cited at p. 67]

[150] K. E. Batcher, "Sorting networks and their applications," in *Proceedings of the April 30–May 2, 1968, spring joint computer conference*. ACM, 1968, pp. 307–314. [cited at p. 67, 78]

[151] D. E. Knuth, *The art of computer programming: sorting and searching*. Pearson Education, 1998, vol. 3. [cited at p. 67, 72]

[152] Y. Erlich and A. Narayanan, "Routes for breaching and protecting genetic privacy," *Nature Reviews Genetics*, vol. 15, no. 6, pp. 409–421, Jun. 2014. [cited at p. 68]

[153] S. Simmons and B. Berger, "Realizing Privacy Preserving Genome-wide Association Studies," *Bioinformatics*, p. btw009, Jan. 2016. [cited at p. 68]

[154] F. Yu, S. E. Fienberg, A. B. Slavkovi, and C. Uhler, "Scalable privacy-preserving data sharing methodology for genome-wide association studies," *Journal of Biomedical Informatics*, vol. 50, pp. 133–141, Aug. 2014. [cited at p. 68]

[155] B. Anandan and C. Clifton, "Laplace noise generation for two-party computational differential privacy," in *2015 13th Annual Conference on Privacy, Security and Trust (PST)*, Jul. 2015. [cited at p. 69]

[156] A. Narayan and A. Haeberlen, "DJoin: Differentially Private Join Queries over Distributed Databases," in *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 149–162. [cited at p. 69]

[157] K. Jiang, D. Shao, S. Bressan, T. Kister, and K.-L. Tan, "Publishing trajectories with differential privacy guarantees," in *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*. ACM, 2013, p. 12. [cited at p. 69]

[158] N. Mohammed, D. Alhadidi, B. C. M. Fung, and M. Debbabi, "Secure Two-Party Differentially Private Data Release for Vertically Partitioned Data," *IEEE Trans. Dependable Secur. Comput.*, vol. 11, no. 1, pp. 59–71, Jan. 2014. [cited at p. 69]

[159] S. Kotz, T. Kozubowski, and K. Podgorski, *The Laplace distribution and generalizations: a revisit with applications to communications, economics, engineering, and finance*. Springer Science & Business Media, 2012. [cited at p. 69]

[160] B. Eisenberg and R. Sullivan, "Why is the sum of independent normal random variables normal?" *Mathematics Magazine*, vol. 81, no. 5, pp. 362–366, 2008. [cited at p. 69]

[161] N. Pippenger, "Selection networks," *SIAM Journal on Computing*, vol. 20, no. 5, pp. 878–887, 1991. [cited at p. 72]

[162] C. Liu, X. S. Wang, K. Nayak, Y. Huang, and E. Shi, "ObliVM: A Programming Framework for Secure Computation," in *2015 IEEE Symposium on Security and Privacy (SP)*, May 2015, pp. 359–376. [cited at p. 73, 78]

[163] K. Nayak, X. S. Wang, S. Ioannidis, U. Weinsberg, N. Taft, and E. Shi, "GraphSC: Parallel Secure Computation Made Easy," in *2015 IEEE Symposium on Security and Privacy (SP)*, May 2015, pp. 377–394. [cited at p. 73]

[164] A. R. Wood *et al.*, "Defining the role of common variation in the genomic and biological architecture of adult human height," *Nature Genetics*, vol. 46, no. 11, pp. 1173–1186, Nov. 2014. [cited at p. 73]

[165] S. Simmons and B. Berger, "One Size Doesn't Fit All: Measuring Individual Privacy in Aggregate Genomic Data," in *2015 IEEE Security and Privacy Workshops (SPW)*, May 2015, pp. 41–49. [cited at p. 73]

[166] Y. Chen, B. Peng, X. Wang, and H. Tang, "Large-Scale Privacy-Preserving Mapping of Human Genomic Sequences on Hybrid Clouds." in *NDSS*, 2012. [cited at p. 74]

[167] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption," Cryptology ePrint Archive, Tech. Rep. 144, 2012. [cited at p. 82]

Index

Zhicong Huang

Contact Information

Postal address:
EPFL IC ISC LCA1,
BC 266 (Batiment BC)
Station 14
CH-1015 Lausanne
Switzerland

Work Telephone: +41216933697
Mobile: +41786729465
E-mail: zhicong.huang@epfl.ch

Summary

Researcher in security and privacy, applied cryptography, network and communication security, big data analysis and security, secure computation, trusted computing (e.g., Intel SGX), privacy-preserving machine learning

Education

École Polytechnique Fédérale de Lausanne, Switzerland **Sep., 2012 – June., 2018**
Doctoral Program in Computer, Communication and Information Sciences

- PhD Thesis: On Secure Cloud Computing for Genomic Data: From Storage to Analysis
- Advisors: Jean-Pierre Hubaux (jean-pierre.hubaux@epfl.ch)
  Jacques Fellay (jacques.fellay@epfl.ch)
- Expected graduation date: March 2018

Peking University, P.R.China **Sep., 2008 – Jul., 2012**
B.S., Computer Science

Honours and Awards

Distinguished Student Paper Award (36th IEEE Symposium on Security and Privacy), 2015
SAMSUNG Scholarship for Academic Excellence, Peking University, 2010
Wusi Scholarship for Academic Excellence(Twice), Peking University, 2009-2011

Work Experience

Microsoft Research Intern **Jun., 2017–Sep., 2017**

- Improved performance of homomorphic encryption library; Designed and implemented a new library for private set intersection
- Machine learning on encrypted data: logistic regression, deep learning

Projects

Distributed cluster analysis of medical data **Apr., 2016 – Mar. 2018**
in *LCA1*, EPFL (in collaboration with *Sophia Genetics*)

- ✓ Summary: Mixture modelling and encryption, Intel SGX, Differential privacy
- ✓ C++ project

Security and Compression of Aligned Sequencing Data **Aug., 2013–Jul., 2015**
in *LCA1*, EPFL (in collaboration with *Sophia Genetics* and *Stanford University*)

- ✓ Summary: Genomic data encryption, compression, and selective retrieval
- ✓ Java project
- ✓ Two papers published, two patents granted
- ✓ Github: https://github.com/acs6610987/secram

GenoGuard **Nov., 2013– Dec., 2015**
in *LCA1*, EPFL (in collaboration with *Jacobs Institute, Cornell Tech*)

- ✓ Summary: Robust password-based encryption for genomic data
- ✓ Python project
- ✓ One paper published in IEEE S&P with distinguished award
- ✓ Github: https://github.com/acs6610987/GenoGuard

CredibleWeb **Sep., 2012–Jan., 2013**
in *LSIR*, EPFL

- ✓ Summary: Crowdsourcing Evaluation of Web Credibility
- ✓ Github: https://github.com/acs6610987/CredibleWeb
- ✓ Google appengine and python webapp2 framework
- ✓ One extended abstract published in CHI

A Drowsiness Detection System on Android Smart Phones **Dec., 2011–July., 2012**
in *Networking Lab*, Peking university

- ✓ Bachelor thesis (award for excellence of bachelor thesis)

✓ Android project
✓ A machine learning project using deep forward networks

Publications

SQC: Secure Quality Control for Meta-Analysis of Genome-Wide Association Studies
Z. Huang, H. Lin, J. Fellay, Z. Kutalik, J.-P. Hubaux
Bioinformatics, 2017 Aug 1, 33(15): 2273-2280.

Unlynx: A Decentralized System for Privacy-Conscious Data Sharing
D. Froelicher, P. Egger, J. S. Sousa, J. L. Raisaro, Z. Huang, C. Mouchet, B. Ford, J.-P. Hubaux
Proceedings on Privacy Enhancing Technologies, pp. 232-250, vol. 4, 2017.

Efficient and Secure Outsourcing of Genomic Data Storage
J. S. Sousa, C. Lefebvre, Z. Huang, J. L. Raisaro and C. Aguilar-Melchor, M.-O. Killijian, J.-P. Hubaux
BMC Medical Genomics, vol. 10, p. 46, 2017.

A Privacy-Preserving Solution for Compressed Storage and Selective Retrieval of Genomic Data
Z. Huang, E. Ayday, H. Lin, R. S. Aiyar, A. Molyneaux, Z. Xu, J. Fellay, L. M. Steinmetz, J.-P. Hubaux
Genome Research, vol. 26, pp. 1687-1696, 2016.

Differential Privacy with Bounded Priors: Reconciling Utility and Privacy in Genome-Wide Association Studies
F. Tramèr, Z. Huang, E. Ayday, J.-P. Hubaux
22nd ACM Conference on Computer and Communications Security (CCS), 2015 .

GenoGuard: Protecting Genomic Data against Brute-Force Attacks
Z. Huang, E. Ayday, J. Fellay, J.-P. Hubaux, and A. Juels.
36th IEEE Symposium on Security and Privacy (S&P 2015), San Jose, CA, USA, May 2015. (**Distinguished student paper award**)

Quantifying Genomic Privacy via Inference Attack with High-Order SNV Correlations
S. S. Samani, Z. Huang, E. Ayday, M. Elliot, J. Fellay, J.-P. Hubaux, Z. Kutalik.
2nd International Workshop on Genome Privacy and Security (in conjunction with IEEE S&P 2015), San Jose, CA, USA, May 2015.

Practical Solutions for Protecting Individual Genomic Privacy
J. Fellay, J. L. Raisaro, Z. Huang, M. Humbert and E. Ayday et al.
American Society of Human Genetics (ASHG), San Diego, CA, USA, 2014.

CredibleWeb: A Platform for Web Credibility Evaluation
Z. Huang, A. Olteanu, K. Aberer.
CHI'13 Extended Abstracts on Human Factors in Computing Systems, Paris, France, 2013.

Skills

Programming Languages

- Most experienced: Java, C/C++, Python
- Used in the past: SQL, JavaScript, Matlab, PHP, HTML