Privacy Threats and Cryptographic Solutions to Genome Data Processing

Chibuike Ugwuoke

Privacy Threats and Cryptographic Solutions to Genome Data Processing

Privacy Threats and Cryptographic Solutions to Genome Data Processing

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus Prof.dr.ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Friday 21 May 2021 at 15:00 o' clock

by

Chibuike Innocent Ugwuoke

Master of Science in Information Security
University College London, United Kingdom,
born in Nsukka, Nigeria.

This dissertation has been approved by the promotors,

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof.dr.ir. R.L. Lagendijk,	Delft University of Technology, promotor
Dr. Z. Erkin,	Delft University of Technology, co-promotor
Independent members:	
Prof.dr.ir. M.J.T. Reinders,	Delft University of Technology
Prof.dr.ir. C. Vuik,	Delft University of Technology
Prof.dr. M. Dumontier,	Maastricht University
Prof.dr. M. Conti,	University of Padua, Italy
Prof.dr. S. Katzenbeisser,	University of Passau, Germany

Keywords: GWAS, genome privacy, privacy enhancing techniques, multi-party computation, machine learning.

Printed by: IPSKAMP Printing

Cover Design: Chibuike Ugwuoke

ISBN 978-94-6421-363-8

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

CONTENTS

SUMMARY

The genome is the blueprint of life and has a detailed genotype and phenotype description of any organism. This in itself attributes sensitivity to genetic data, be it in the biological or electronic format. The possibility of sequencing the genome has opened doors to further probing of the data in its electronic form. Post sequencing of the biological genome sample, the electronic genome is stored, processed, and transmitted for variety of purposes including but not limited to Medicare, research, solving crimes and entertainment. However, due to the sensitivity of the genome data, security and privacy of the electronic data is considered to be imperative.

Owing to the privacy and security concerns associated with sharing genome data with third-party entities for processing, various secure and privacy-preserving solutions have been considered. Such scenarios include, a researcher obtains research data which includes genome of individuals, or when a healthcare institution outsources the genome of its patients to a cloud environment for storage and processing. In all of these scenarios, it is important that the utility (accuracy and efficiency) of the data is maintained while preserving privacy (confidentiality and unlinkability) simultaneously.

In this thesis, we focus on maintaining data utility when processing electronic genome data as well as preserving the privacy of the individuals whose data are analysed. We employ privacy enhancing techniques such as secure multi-party computation and homomorphic encryption to existing problems and develop provably secure cryptographic protocols that are fit for purpose for each scenario.

SAMENVATTING

Het genoom is de blauwdruk van het leven en bevat een gedetailleerde beschrijving van het genotype en fenotype van elk organisme. Dit gegeven maakt dat genetische gegevens zeer gevoelige informatie is, of dit nu in biologische of elektronische vorm is. De mogelijkheid om het genoom te sequentieren heeft deuren geopend voor het verder onderzoeken van de gegevens in hun elektronische vorm. Na sequentiebepaling van het biologische genoommonster wordt het elektronische genoom opgeslagen, verwerkt en verzonden voor verschillende doeleinden, inclusief maar niet beperkt tot medische doeleinden, onderzoek, het oplossen van misdaden en vermaak. Vanwege de gevoeligheid van de genoomgegevens worden beveiliging en privacy van de elektronische gegevens echter als noodzakelijk beschouwd.

Vanwege de privacy- en beveiligingsproblemen die gepaard gaan met het delen van genoomgegevens met externe entiteiten voor verwerking, zijn verschillende beveiligings- en privacybeschermende oplossingen overwogen. Dergelijke scenario's zijn onder meer: een onderzoeker verkrijgt onderzoeksgegevens die het genoom van individuen bevatten, of wanneer een zorginstelling het genoom van zijn patiënten uitbesteedt aan een cloudomgeving voor opslag en verwerking. In al deze scenario's is het belangrijk dat zowel de bruikbaarheid (nauwkeurigheid en efficiëntie) van de gegevens, als ook de privacy (vertrouwelijkheid en onkoppelbaarheid) behouden blijft.

In dit proefschrift richten we ons op het behouden van de bruikbaarheid van gegevens bij het verwerken van elektronische genoomgegevens en op het beschermen van de privacy van de individuen van wie de gegevens worden geanalyseerd. We gebruiken technieken om de privacy te verbeteren, zoals secure multi-party computation en homomorphic encryption voor bestaande problemen, en ontwikkelen aantoonbaar veilige cryptografische protocollen die geschikt zijn voor elk scenario

1

INTRODUCTION

Investigating the human genome in the hopes of understanding its basic compositions, history and evolution can be considered as a daring feat by our species to unlock the secret to what makes us exceptional but vulnerable to simple diseases. In fact, the successes recorded while exploring the genome has presented humans with one major outcome. This being the harvest of explosive information previously locked within the human genome, but surprisingly tailgated by the unplanned privacy-invasive potentials of investigating the genome of a non-consenting individual. This thesis focuses on providing provable solutions via cryptography, that allow for humans to enjoy the utilities that accompany genome data research while preventing the security/privacy-invasion of participating members.

1

1.1. GENOME DATA PROCESSING

A comprehensive set of any organism's deoxyribonucleic acid (DNA) is known as that organism's genome [1]. The human genome has been accorded huge attention within the last two decades, and this is a direct consequence of the revolutionary corner turned by the research community with the realisation of whole genome sequencing [1–4]. Therefore, the possibility of having an electronic copy of the genome (*in silico* genome) has not only become common, but interestingly, is available for quite affordable units of time and money [5–7]. With digital copies of genomes easily obtainable, genome data processing (GDP) becomes inevitable, and has been commonly adopted both in research and industry. This adoption owes it to the rich information embedded in the genome, which is relevant for bio-research [8]. Some of the motivations for adopting GDP includes its potentials: personalised healthcare, disease predisposition, patients response to treatments and maybe early discovery and cure for diseases such as cancer [9–12]. Due to the increasing popularity of electronic copies of genome data, various domains of expertise in research and industry have been involved in one or more aspects of GDP, including faster sequencing techniques[13] and genotype-trait association discoveries [14, 15]. Interestingly, the idea of an integrated genomic and proteomic security protocol has been proposed and patented, towards the use of genomes for the construction of authentication and encryption suit [16].

1.1.1. PROPERTIES OF THE GENOME

There are properties of the genome that make it peculiar and attractive for interested parties who wish to investigate and utilize the genome data. Some of these properties include:

- *Immutability:* the genome of an individual is stable and does not change over time. This means that in the event that a person loses his genome to an adversary, it is not possible to make changes to obtain a new set of genome.
- *Uniqueness:* the genome is considered the blueprint of life. It is unique for every individual, even for identical (monozygotic) twins.
- *Shared properties:* even though the genome is unique to the sequenced individual, information about close relatives of the individual is contained in the genome. This means that siblings carry around a genetic information of one another and this widens the surface of attack for an adversary.
- *Predictive power:* the genome houses a great deal of information. It can be used to predict phenotypes which are not currently expressed on the individual.
- *Longevity:* the usefulness of the genome persist even after the direct owner who was sequenced is long dead. It means that information obtained from a deceased individual can always be useful for relatives who are alive or other purposes.

1.1.2. RELEVANCE OF GENOME DATA PROCESSING

In the context of this thesis, genome data processing describes those activities that involve the digital version of the genome all through the lifespan of the genome. A typical

life cycle of a digital genome includes four major phases:

1. *Sequencing:* the process of reading a biological sample of an individual, and reproducing an electronic representation of the the biological genome.

2. *Storage:* the ability to securely store the electronic genome data, for efficient use of the data. This is done immediately after sequencing the biological sample of the genome owner.

3. *Processing:* of the genome either, independently, as a part of a large dataset or in combination with other non-genomic dataset.

4. *Publishing:* results that might have included genome data as input data during processing.

On account of the list above, we encapsulate our concept of GDP to represent any operation that involves the electronic genome which can be categorized into either *sequencing*, *storage*, *processing* or *publishing* phase.

Sequencing is the initialization phase of the entire life cycle of a typical electronic genome. This phase is exclusively handled by the experts from the biology domain, and often relies on the integrity of the biologist. The implication being that there is very little that can be done by means of security or privacy to protect possible abuse in this phase. In fact, this thesis is written with the assumption that the *sequencing* phase is handled by a party who does not have the incentive to misrepresent the genome of the customer, which is supported by the ethics guidelines of handling health data [17–19].

The *storage* phase refers to the efficient methods for securely storing the genome data in such a way that the genome data can be easily accessed for use. The cost of storing one individual's complete genome is estimated at 150 gigabytes of space [20], making it non-trivial to access and use the data. Such usage might include viewing, insertion, deletion, editing, updating, transfer or sharing the digital genome.

The *processing* phase represents a broader spectrum of operations, which can include: i) Simple analysis of the genome data against a *reference genome*. ii) The use of genome data alongside a pool of other genome data for an association study involving thousands of participants. iii) Using genome data independently, to compute a desirable status for the genome owner. The operations in the *processing* phase typically requires mathematical, statistical and machine learning functions/algorithms. For example, a relatively simple statistical function like Pearson Goodness-Of-Fit test can be computed using genome data [21, 22], the result is thereafter utilized in making critical decisions such as the significance of a gene to a disease [22]. Other complex operations such as logistic regression and deep learning can be observed in [23–25], where machine learning is utilized in combination with genome data, and in developing machine learning models that are of interest to researchers.

In the *publishing* phase, the results obtained from any of the previous phases can be made public to either a select group of individuals or the general public, depending on what the objective of the entity that publishes the result [26]. An example would include publishing a newly discovered association between genes and diseases, or just reporting the status of a clinical test result to a patient.

Therefore, one can argue that genome data processing is commonly dependent on machine learning algorithms. In fact, machine learning has remained an integral part of most operations that involve the genome. This is because the huge amount of data very typical of genome research begs for optimal, fast, and complex algorithms that would identify complex relationships, and make sense of the large data in order to reduce time which might be spent investigating uninteresting data [20, 24]. The adoption of machine learning by computational biologist and bio-statisticians in processing genome data offers the robustness and big data processing ability, which would otherwise be unattainable if traditional data-processing applications software were to be used [27].

With GDP, the possibilities are boundless [16, 28, 29], with seemingly as many positive contributions as there are perceived threats. On one hand, through the research on genome data, it is expected that personalized medicine will be improved such that every individual receives medications that are tailored to their genomic composition as opposed to that of a population [30]. Also, individuals could test for their predisposition to diseases by examining their genome or that of their relations, using genome data processing [12]. This will lead to early diagnosis, and eventually preventive medicare, rather than the reactive medicare commonly obtainable. This list of positive contributions includes test for ancestry information, paternity or maternity determination, patients response to treatment, and better understanding of diseases like cancer[8], just to mention a few. On the other hand, the possibility of an individual exposing his genome data to the public raises realistic privacy concerns. For instance, consider a participant *Alice*, who had contributed his genome data for a large cancer research study, if he is publicly identified. If *Alice* were to be re-identified publicly to have participated in the study, notwithstanding if she was grouped in the case or control group of the study, she is very likely to deal with the abuse of privacy-inversion.

The importance of genome data processing cannot be overemphasised as it touches multiple domains, hence, widening the circle of stakeholders to be considered during the lifespan of a genome data. Genome data processing spans its relevance to individuals, patients, medical personnel, data scientists, bioinformaticians, biostatisticians, bioresearchers, commercial companies, pharmaceutical companies, lifestyle coaches, environmental researchers [6]. Each stakeholder may have different interests in GDP as the next stakeholder. Sometimes, these interests tend to conflict thereby requiring novel ideas on how to resolve such conflicts without disrupting the utility of the data [11]. A depiction of such a conflict can be observed in the scenario where a researcher wishes to conduct a machine learning study using the genome data of multiple participants, and these participants care about their privacy and wish to only provide the data in a protected form that cannot be abused by the researcher [31].

Summarily, the plummeting in cost of sequencing the genome has brought about the proliferation of *in silico* genome and genome data processing. Thereby, inviting previously non-interested parties into the discourse by ways of intensive research, novelty in data analytic techniques, and of course offering genome data processing services for profit [11].

The rest of this Chapter is structured as follows: first we introduce the stakeholders commonly found in literature and industry, in order to establish their roles. Then, we present the research motivation for investigating genome data processing, followed

by the business motivation explaining its wide attraction in the industry. Furthermore, we present the concept of genome data as a service, followed by the associated privacy threats. Thereafter, existing solutions to the identified threats are presented alongside the open problems to be addressed in this thesis. From the identified open problem, a problem statement is formulated for this thesis with a clearly stated research question to be addressed. Finally, the contributions of this thesis are present and quickly followed with an outline for the rest of the chapters.

1.2. RESEARCH AND MEDICAL INCENTIVES

The genome has been studied in medical research and related fields, and the success of whole genome sequencing has accelerated genome related research since the turn of the 21st century [6, 32]. Consequently, the advent of digital genome has aided the research community in making significant progress due to the comparative ease of analysing the digital genome as against the use of biological samples [33–35]. As a result, various studies conducted using the genome have impacted medicine in the following ways:

1. Providing scientists with significant understanding of the basic building blocks of life [36].

2. Aided in investigating relationships between genes and phenotypes [37]. With the aid of genome-wide association study (GWAS), researchers have been able to establish relationships associations between genetic variations and specific phenotypes [38].

3. Improved our knowledge of diseases and their causes [39].

4. Helped in study of patients response to treatment and personalized medicine [40].

5. Stands in the frontier for the development of preventive medicare [41].

GlaxoSmithKline, a pharmaceutical giant, recently announced that DNA results from the 5 million customer base of 23andMe [42] will be used to design new drugs. This, suggests that the interest in genome data processing persist and the possibilities are convincing, even for big pharmaceuticals [43]. In support of the research toward promoting privacy for genome data, a body known as *integrating data for analysis, anonymization, and sharing* (iDASH) [44], whose focus is on privacy-preserving algorithms and solutions for data sharing, have continued to push the boundaries of unresolved problems in genome data processing research and algorithms. One way iDASH does this is by announcing open problems as challenges yearly, and accepting contributions from all over the world in hope of closing some research gaps, solving previously open problem, and realizing more secure, privacy-preserving, and efficient solutions to genome data processing.

1.3. FINANCIAL AND BUSINESS MOTIVATION

Ever since the successful completion of whole genome sequencing [1] about two decades ago, the cost for sequencing the genome has continued on a downward trend, making it even more affordable for people, see Fig 1.1. While it can be argued that more individuals

now seek to use GDP services due to the sheer affordability of these services, some group of people equally choose to patronize the companies out of curiosity and fun.

As a result, investors have taken advantage of the possibility of genome sequencing to offer genome related services. These commercial entities have clearly taken advantage of the existing demand, by commodification of the genome and genome related services. Hence, offering services such as sequencing, ancestry testing, disease susceptibility testing and many more for a token, to a readily available customer base. This has led to the proliferation of *in silico* genome, while also pushing the boundaries of research towards exploring other uses for the genome. It can be said that these companies invest to offer commercial services to individual in a market that has naturally found its niche.

For as low as $100, commercial companies such as Helix, MyHeritage and 23andMe state that they are able to unlock your genome and deliver health and ancestry services directly to customers [42, 45, 46], without the intervention of your medical doctor. GlaxoSmithKline are reported to have invested $300 million in 23andMe, even though 23andMe are already valued at about $1.75 billion [47, 48]. As of December, 2019, 23andMe recorded an estimated annual revenue of $475 million, with a total funding of $877.7 million as against its total funding of $27.8 million in 2009 [49]. The numbers clearly support the argument that genome data processing is a financially viable area for investment. In fact, what the commercial companies such as 23andMe and Helix have done is to offer genome data processing as a Service (GDPaaS). We mention that existing market models that are found in the wild at the time of writing this thesis is such that the commercial companies play the roles of *Sequencer* and *Storage and Processing Unit*.

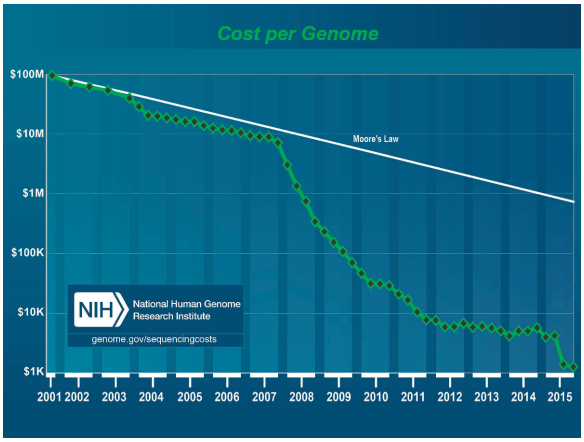

Figure 1.1: The plummeting in cost of genome sequencing

1.3.1. GENOME DATA PROCESSING-AS-A-SERVICE

Many of the customers who patronise the *storage and processing unit* are *Data Owners*, who seek these services for their personal interests. However, *Medical Personnel* and *Researchers* equally benefit from these services. For instance, a medical doctor might require a specific genome related test for a patient. Where such tests are not readily

obtainable in their hospitals, the medical doctor would refer the patient to one of the commercial companies. For more complex processing, where huge data is required or specialized computing hardware and software are needed, these computationally and time intensive computations are better outsourced to a *Storage and Processing Unit* that offer the corresponding services. This scenario of transferring data processing to a third party can be qualified as outsourced computation.

Outsourced computation allows for a computationally limited client to delegate computationally demanding functions to a highly resourced party such as a cloud infrastructure. In this scenario, the cloud infrastructure represents the company offering GDPaaS, and is expected to compute the functions correctly and return result to the client. A list of services offered under GDPaaS includes the following:

- **Sequencing:** This is usually the first of many services purchased by customers. This typifies the contact between the *Data Owner* and the *Sequencer*. The biological sample presented by the *Data Owner* is sequenced to obtain a digital copy of the genome. The result of the sequencing step can be a suit of single nucleotide polymorphism (SNP) of the customer. SNPs are genetic markers and are generated by comparing each locus of the customer's genome against a set of reference genomes [50], and the result is a fraction of the customers genome. It is estimated that every individual has about 4 to 5 million SNPs in their genome [51]. Another result of the sequencing step could also be the whole genome of the customer, this requires that all the nucleotide base pairs of a customer is obtained and this numbers about 3 billion base pairs.

- **Storage:** After sequencing a *Data Owner*'s genome to obtain either SNP values or the complete genome, the storage and management of the electronic genome can be offered as a service to the customers. This is because a customer might not have the hardware needed to manage these large dataset especially if it is an individual. In some cases where special software are required to access the data, even large medical institutions are happy to outsource this function to the commercial company. Since they are not expected to have the expertise and resources required for the management of genome data of multiple people.

- **Analysis:** Sequencing the genome is usually not the end goal, neither is storage. The aim is generally to analyse the data for one purpose or another, be it for population study or a medical diagnosis. The analysis varies depending on the need of the client, which means that the companies provides a suit of analyses as part of it GDPaaS, and the customers only pay for the services they seek. The analysis services can also vary in complexity, from basic comparison that reports similarities with another genome, to complex models that takes a customer's genome as input and outputs the degree of the customer's susceptibility to a disease of interest. Genome wide association study (GWAS) can also be outsourced by institutions to the commercial companies. GWAS is typically computationally intensive because of the complex statistical function and machine learning models involved, and also it utilizes huge dataset [35, 38, 52]. This makes GWAS a preferable candidate

of GDPaaS, since it is cheaper for institutions to outsource these operations rather than spend money and time in building an in-house team with the requisite expertise.

Inasmuch as correctness is an important criteria for cloud infrastructure during outsourced computation, an ever present concern is that of protection of the outsourced data from abuse, especially in the face of critically sensitive dataset like the genome.

1.4. PRIVACY CONCERNS IN GENOME DATA PROCESSING

The genome is a highly identifying data, with the ability to uniquely identify any individual, making it necessary to seek the consent of its owner before using them [10, 53, 54]. With the proliferation of gathering genomic dataset and the continuous growth of databases of genomic data held at diverse locations across the globe, it raises serious privacy concerns amongst researchers, participants and even governments. The possible catastrophe of the misuse of such an important dataset, should better be imagined [55]. Therefore, it has necessitated the search for procedures that would allow viable research involving genomic data, while not compromising the privacy of participants [56].

The threats towards abuse of genome data is well documented [57–61], and efforts have been made to address some of these threats. As an example, in 2013 Gymrek et al. demonstrated the possibility of re-identifying participant or their close family members by analyzing chromosome sequences which are readily available on public genetic genealogy websites [62]. Thereby, de-anonymizing the participants and threatening their privacy. At least, this shows that simple anonymization techniques are not enough to protect genome data, and in extension the privacy of the owners. A comprehensive suit of solution for protecting genome data requires contributions which is multi-disciplinary. This requires dedicated and collaborative efforts from, social sciences, legal, and information technology to properly regulate the use of genetic data [11, 26, 56]. The reason for promoting a multi-disciplinary solution include:

- The protection of genome data while seeking to preserve its utility is a difficult problem that requires a non-trivial solution [11, 63]. Its property to uniquely identify any individual makes it really attractive for adversaries, and these adversaries can target the weakest link in the life cycle of the genome. And this includes creating awareness on the possible risks of exposing genome data.
- Because the genome originates from a biological sample, it becomes almost impossible to apply only information technology protection to its life cycle. This is the reason laws such as the Genetic Information Nondiscrimination Act of 2008 have been established to assist in genome data protection [64, 65].
- A genome data obtained for the use of specific purpose, can equally be used for a variety of other purposes. This opens the door for potential abuse, hence, calling for more elaborate and comprehensive approaches for protecting genome data [66].

A complete boycott of analyses of genome data due to the privacy threats would be counterproductive, because the research community and humans would lose out on associated innovations. It boils down to the trade-off between the privacy versus the utility

of genome data processing that is acceptable for relevant stakeholders [31]. This thesis therefore provides an information technology solution to privacy-threats in genome data processing, through the use of provable cryptographic techniques.

1.4.1. THREATS

The availability of digital genome data to the general public has resulted in the sprouting of terms such as “cybergenetics", which in turn has pushed for more conversations on the awareness of genome data processing. The conversation about processing the genome often differs depending on whom the audience is currently paying attention to. For example, while participants of a large research would appreciate not revealing their identity in a dataset used for a cancer study, the commercial company who offers services like tests for disease susceptibility wishes to protect the algorithms they deploy in order to protect trade secrets.

Concerns for potential abuse of genome data are rife, and this reaffirms the call for protection of genome data [26, 57, 61]. The entire genome data processing community share one concern or another that requires protection of identified assets.

- **Re-identification:** Participants of a large dataset used for a research study would wish that they are not identified as having partook in such studies and would wish for their identity to be confidential and anonymous [31, 57, 62].

- **Stigmatization:** An individual who undertakes a disease susceptibility test using genome data, expects that the results of such tests are accurate and prefers not to be stigmatized for having requested the test for a particular disease. Another example can be seen in a scenario where institutions share unprotected data with other institution when jointly performing new studies. The institutions would insist on the privacy and confidentiality of participants in their dataset [67, 68]. Hence, they will demand privacy and security measures consistent with the request of the participants.

- **Discrimination:** An individual does not want to be discriminated by an insurance company or to be attributed high premium on the basis of his genetic composition [31, 67].

- **Confidentiality:** Finally, as an addendum, we have considered the security risks associated with the commercial companies where it may be relevant. For instance, a commercial entities may wish to protect their proprietary algorithms when used to render services to the public. Therefore, they insist on the confidentiality of their algorithm since it can be considered a trade secret, with huge financial implications if it were to be leaked.

Depending on the peculiarity of the scenario, threat models differ and are often determined by parties and the interactions amongst such parties. The listed threats above can be summarised into the desired security properties relevant in the genome data processing ecosystem. These properties are *integrity*, *confidentiality*, and *privacy*.

1

1.4.2. EXISTING SOLUTIONS AND OPEN PROBLEMS:

Although some common scenarios have been discussed in literature are and often followed by proposed solutions, there are still research gaps even in some of the proposed solutions [57]. We introduce the scenarios we attempt to address and the open problems associated with them.

- **Scenario 1:** Consider a scenario where a research institution sources genome data from volunteers, and wishes to outsource the GWAS to an untrusted cloud infrastructure. In this case, the volunteers are happy to partake in the study, with the caveat that they cannot be re-identified from the dataset nor from the published results. This problem has been presented with different information technology solutions over the years. A commonly adopted approach has been the introduction of special encryption algorithms known as homomorphic encryption, in order to provide confidentiality and equally allow for basic mathematical operations to be carried out on the ciphertext [22, 69]. However, homomorphic encryption comes at a huge costs and can equally be limited. This means that where the function to be computed requires more than additions, subtraction, and scalar multiplications, the costs of a homomorphic solution accelerates from very expensive to practically impossible. Therefore, there is still the need for privacy-preserving but efficient solutions to this problem.

- **Scenario 2:** In this scenario, a customer already owns electronic copy of his genome, and wishes to utilize a GDPaaS for testing his susceptibility for a disease. If the algorithm for computing the susceptibility testing were developed by the commercial company and therefore treated as a trade secret, the company will be interested in protecting the algorithm. This means the customer may need to transfer the genome data to the company in order to benefit from the services. Here, we see that the data privacy requirement of the customer tends to conflict with the algorithm's confidentiality requirement expected by the customer. Existing solutions [70, 71] have proposed the use of homomorphic encryption to solve this problem, while others have recommended the use of secret sharing technique. Notwithstanding the contributions of these proposals, they commonly suffer from inefficiency and sometime out-rightly relaxing the security requirements, therefore making them unsuitable for deployment in the wild.

- **Scenario 3:** Let us assume that two or more research institutes wish to jointly conduct a study using genome data. The aim of the study is to compute a collaborative machine learning model using contributed genome dataset. Also, the parties do not wish to share their data unprotected, but are happy to share the models resulting from the study so long as the original data cannot be reconstructed from the published models. At the time of writing this thesis, this scenario remained a challenge both in industry and academia, and an efficient, provable, and privacy-preserving solution remains an open problem, so much so that it appeared as a task in the 2019 iDASH challenge.

1.5. PROBLEM STATEMENT

The primary aim of this thesis is to proffer provable information technology related solutions that allow for the protection of genome data. And, at the same time encourage stakeholders who need genome data to have regulated access while they continue to enjoy the utility afforded by genome dataset.

As part of a technological contribution to protection methods in genome data processing, we propose solutions to the privacy and security threats inherent in the genome data processing ecosystem. We tailor our proposals to concentrate on adopting cryptographic primitives to develop provably-secure and privacy-preserving protocols for genome data processing. However, for problems which have existing solutions or proof of concepts that address our identified problems, we seek to improve on those solutions, but where such improvements are not feasible, and create novel protocols where improvements are not realistic.

Primitives such as homomorphic encryption, secret sharing and other privacy preserving technologies will be considered as means of computing the underlying algorithms already used within genome data processing sphere. We do not pursue the invention of novel algorithms for genome data processing, but we hope to wrap the existing techniques with a privacy preserving primitives, in order to allow for the same computations with these genomic dataset in privacy safe manners. Finally, we anticipate that the privacy preserving results must not be significantly worse off than is currently obtainable in the unprotected versions.

With respect to the identified open problems in 1.4.2, we formulate the following research question.

For every party in our defined setting/scenario of genome data processing, how can we utilize cryptographic primitives to provide provable privacy and security for all identified assets within the threat model, and still realize the utility of the protocol as a relatively efficient alternative?

There are three core objectives of this research question, and we summarize them in the following sub-questions:

1. **Privacy and Security:** How do we design protocols with the aim of providing provable privacy and security guarantee, and equally optimize the computational, communication, and storage cost of realizing such a protocol?

2. **Acceptable Accuracy:** How can we preserve the utility of genome data services even in the protected domain, such that our privacy-preserving variants can replicate the accuracy obtainable in the non-protected variants of the protocols.

3. **Performance:** How can we realize privacy-preserving protocols that are not bedeviled by poor performances, but rather enhance performance by supporting efficient storage, acceptable communication complexity, and practical computational efficiency.

1

1.6. CONTRIBUTION OF THIS THESIS

This thesis draws attention to the security and privacy threats existing in the ecosystem of genome data processing. Then, it addresses some of the identified privacy and security threats by introducing provable, secure, and privacy-preserving protocols as proof-of-concepts. Listed below are some of the core contributions of this thesis:

1. We propose one of the first privacy-preserving, non-interactive and parallelized *Machine Learning-as-a-Service* protocol, that is able to offer learning on labelled dataset, model generation, classification, and statistical significance computation on encrypted dataset. This proposal was submitted as a solution to iDASH 2018 challenge, where it was ranked in the 6^{th} position with respect to accuracy and resource optimization. The details of this proposal is presented in Chapter 5.

2. In this thesis, we show how machine learning can still be efficiently performed over protected dataset, and in some cases, even the machine learning algorithms can be generated and used for data classification even while preserving privacy for the model. Our work in Chapter 4 demonstrates this contribution.

3. We show that some genome data processes and algorithm can be executed in the privacy-protection domain without having to lose significant efficiency nor accuracy of results. In fact, we argue with a proof of concept that in certain scenarios, with a smart choice of machine learning algorithm, some privacy-preserving solutions can outperform some unprotected solutions of the machine learning problem. The solution as described in Chapter 5 was submitted to the iDASH 2019 challenge, where it was ranked 5^{th} in its category.

4. We demonstrate that privacy-preserving solutions can equally be obtained without the use of expensive primitives such as homomorphic encryption. We do this by replacing homomorphic schemes with data masking and obfuscation in order to greatly improve on the speed and resources optimization of the novel protocol. In this work as described in Chapter 4, our approach outperforms an existing state-of-the-art solution by more than 98%.

5. We adapt multiple privacy-enhancing techniques to solve problem that have previously been classified as computationally expensive. By this, we show the feasibility of merging various techniques towards the aim of providing privacy guarantees for sensitive data. Our adaptation of multi-party computation techniques allow us to demonstrate how simulation-based security proofs can be utilized in providing security and privacy arguments even for solutions with integrated techniques. We demonstrate this technique in Appendix A, also this work presents the first division protocol with all encrypted operands.

6. We realize privacy-preserving collaborative machine learning model generation, using private data contributed by various institutions to enhance the usefulness of genome data research through collaboration? The details of this proposal is presented in Chapter 5.

1.7. THESIS OUTLINE

The rest of this thesis is composed of five chapters an appendices. Each chapter introduces one or more independently written papers. Whereas, some of these papers have been peer reviewed and published, some are currently under review, this means these papers have been submitted to conferences for peer review. Due to the fact that all the works presented in the chapters were independently produced publication, it is expected that notations may not be consistent across the entire thesis. However, each paper will introduce its own notation table to aid in understanding the discussed protocols.

CHAPTER 2

In Chapter 2, we introduce the background to the building blocks required in this thesis. Introduction to basic genome data processing computations and algorithms are provided. Also, cryptographic primitives are introduced and discussed, only to the extent that these primitives are relevant to some section of this thesis. Most importantly, we introduce the concrete scenarios that are of interest to us with a brief discussion on existing solutions that have been proposed to address those scenarios.

CHAPTER 3

In Chapter 3 titled “Privacy-Preserving GWAS", we present three independently written papers. First, a literature survey paper on the threats to genome data processing is provided to present an overview of the domain of privacy-preserving solutions to privacy threats in genome data processing. Second, a peer-reviewed paper on privacy-preserving approach to computing linkage disequilibrium is presented. Third, a peer-reviewed paper that addresses a privacy-preserving association study is presented.

CHAPTER 4

Chapter 4 is titled “Post-GWAS Computation on Privacy-Preserving Data", and discusses post-GWAS computations in a privacy-safe solution. Also, the general concept and solution is presented in the form of a peer-reviewed and published paper, which proffers a solution for disease susceptibility testing.

CHAPTER 5

In Chapter 5 titled “Machine Learning on Protected Genome Data", the concept of integrating privacy-preserving machine learning with privacy-sensitive data is extensively discussed. Various solutions are presented in two different papers. Firstly, the problem of outsourcing machine learning models with private genome data is presented in a paper with a protocol called REDACT. REDACT is a response to a track in the iDASH 2018 competition in which that solution was ranked 6^{th} globally. Secondly, another solution called SCOTML, that equally featured in the iDASH 2019 challenge, where it was ranked 5^{th} is presented. SCOTML focuses on collaboratively generating public machine learning models, using privacy-sensitive dataset from mistrusting parties.

CHAPTER 6

In Chapter 6, we conclude this thesis by discussing the successes achieve with respect to our stated objectives. And also, we discuss open problems that have not been address in this thesis, but have rather been left as future work.

APPENDIX A

In Appendix A, we provide a secure multiparty computation protocol for the aid of performing fixed-point division of encrypted operands. This solution describes the first attempt to compute division using all homomorphically encrypted operands, with prototype implementations for both partial homomorphic and fully homomorphic schemes.

APPENDIX B

In Appendix B, we provide a privacy-preserving solution for collaboratively learning from machine learning models owned by mutually mistrusting parties. This work demonstrates how parties could leverage on already trained models to generate an optimized new global model without having to sacrifice the privacy of their individual dataset.

REFERENCES

[1] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt, *et al.*, *The sequence of the human genome,* science **291**, 1304 (2001).

[2] E. S. Lander, L. M. Linton, B. Birren, C. Nusbaum, M. C. Zody, J. Baldwin, K. Devon, K. Dewar, M. Doyle, W. FitzHugh, *et al.*, *Initial sequencing and analysis of the human genome,* Nature **409**, 860 (2001).

[3] J. Shendure, S. Balasubramanian, G. M. Church, W. Gilbert, J. Rogers, J. A. Schloss, and R. H. Waterston, *Dna sequencing at 40: past, present and future,* Nature **550**, 345 (2017).

[4] 1000 Genomes Project Consortium, *A global reference for human genetic variation,* Nature **526**, 68 (2015).

[5] N. Rusk, *Cheap third-generation sequencing,* Nature Methods **6**, 244 (2009).

[6] E. R. Mardis, *The $1,000 genome, the $100,000 analysis?* Genome medicine **2**, 84 (2010).

[7] E. Pennisi, *Gene researchers hunt bargains, fixer-uppers,* American Association for the Advancement of Science (2002).

[8] M. R. Stratton, P. J. Campbell, and P. A. Futreal, *The cancer genome,* Nature **458**, 719 (2009).

[9] W. S. Bush and J. H. Moore, *Genome-wide association studies,* PLoS computational biology **8**, e1002822 (2012).

[10] NHGRI, *Informed consent,* (October, 2019), `https://www.genome.gov/about-genomics/policy-issues/Informed-Consent` Online; accessed January, 2020.

[11] E. W. Clayton, B. J. Evans, J. W. Hazel, and M. A. Rothstein, *The law of genetic privacy: applications, implications, and limitations,* Journal of Law and the Biosciences (2019).

[12] Y. Smith, *Applications of genomics,* (2019), `https://www.news-medical.net/life-sciences/Applications-of-Genomics.aspx` Online; accessed January, 2020.

[13] E. L. Van Dijk, H. Auger, Y. Jaszczyszyn, and C. Thermes, *Ten years of next-generation sequencing technology,* Trends in genetics **30**, 418 (2014).

[14] S. Horvath, X. Xu, and N. M. Laird, *The family based association test method: strategies for studying general genotype–phenotype associations,* European Journal of Human Genetics **9**, 301 (2001).

[15] J. R. Staley, J. Blackshaw, M. A. Kamat, S. Ellis, P. Surendran, B. B. Sun, D. S. Paul, D. Freitag, S. Burgess, J. Danesh, *et al.*, *Phenoscanner: a database of human genotype–phenotype associations,* Bioinformatics **32**, 3207 (2016).

[16] H. C. Shaw, *Integrated genomic and proteomic security protocol,* (2014), uS Patent 8,898,479.

[17] NHGRI, *Regulation of genetic tests,* (November, 2019), https://www.genome.gov/about-genomics/policy-issues/Regulation-of-Genetic-Tests Online; accessed January, 2020.

[18] A. L. Fairchild and R. Bayer, *Ethics and the conduct of public health surveillance,* (2004).

[19] S. S. Coughlin, *Ethical issues in epidemiologic research and public health practice,* Emerging themes in epidemiology **3**, 16 (2006).

[20] M. Molteni, *Now you can sequence your whole genome for just $200,* (November, 2018), https://www.wired.com/story/whole-genome-sequencing-cost-200-dollars/1 Online; accessed January, 2020.

[21] C. Ugwuoke, Z. Erkin, and I. Lagendijk, *A privacy-preserving GWAS computation with homomorphic encryption,* in *37th WIC Symposium on Information Theory in the Benelux/6th WIC/IEEE SP Symposium on Information Theory and Signal Processing in the Benelux, Louvain, Belgium* (2016) pp. 166–73.

[22] K. Lauter, A. López-Alt, and M. Naehrig, *Private computation on encrypted genomic data,* in *Progress in Cryptology-LATINCRYPT 2014* (Springer, 2014) pp. 3–27.

[23] K. Sikorska, E. Lesaffre, P. F. Groenen, and P. H. Eilers, *GWAS on your notebook: fast semi-parallel linear and logistic regression for genome-wide association studies,* BMC bioinformatics **14**, 166 (2013).

[24] M. W. Libbrecht and W. S. Noble, *Machine learning applications in genetics and genomics,* Nature Reviews Genetics **16**, 321 (2015).

[25] M. K. Leung, A. Delong, B. Alipanahi, and B. J. Frey, *Machine learning in genomic medicine: a review of computational problems and data sets,* Proceedings of the IEEE **104**, 176 (2016).

[26] A. Cambon-Thomsen, E. Rial-Sebbag, and B. M. Knoppers, *Trends in ethical and legal frameworks for the use of human biobanks,* European Respiratory Journal **30**, 373 (2007).

[27] K. Panchal, *3 data science methods and 10 algorithms for big data experts,* (December, 2016), https://datafloq.com/read/data-science-methods-and-algorithms-for-big-data/2500 Online; accessed January, 2020.

[28] A. L. Hopkins and C. R. Groom, *The druggable genome,* Nature reviews Drug discovery **1**, 727 (2002).

[29] W. F. Fricke, D. A. Rasko, and J. Ravel, *The role of genomics in the identification, prediction, and prevention of biological threats,* PLoS biology **7** (2009).

[30] M. S. Boguski, R. Arnaout, and C. Hill, *Customized care 2020: how medical sequencing and network biology will enable personalized medicine,* F1000 biology reports **1** (2009).

[31] E. W. Clayton, C. M. Halverson, N. A. Sathe, and B. A. Malin, *A systematic literature review of individuals' perspectives on privacy and genetic information in the united states,* PloS one **13**, e0204417 (2018).

[32] D. R. Bentley, *Whole-genome re-sequencing,* Current opinion in genetics & development **16**, 545 (2006).

[33] R. Ekblom, C. N. Balakrishnan, T. Burke, and J. Slate, *Digital gene expression analysis of the zebra finch genome,* BMC genomics **11**, 219 (2010).

[34] S. R. Browning and B. L. Browning, *Rapid and accurate haplotype phasing and missing-data inference for whole-genome association studies by use of localized haplotype clustering,* The American Journal of Human Genetics **81**, 1084 (2007).

[35] M. K. Leung, A. Delong, B. Alipanahi, and B. J. Frey, *Machine learning in genomic medicine: a review of computational problems and data sets,* Proceedings of the IEEE **104**, 176 (2015).

[36] J. D. Wall and J. K. Pritchard, *Haplotype blocks and linkage disequilibrium in the human genome,* Nature Reviews Genetics **4**, 587 (2003).

[37] J. P. Pollinger, K. E. Lohmueller, E. Han, H. G. Parker, P. Quignon, J. D. Degenhardt, A. R. Boyko, D. A. Earl, A. Auton, A. Reynolds, *et al.*, *Genome-wide snp and haplotype analyses reveal a rich history underlying dog domestication,* Nature **464**, 898 (2010).

[38] NLM, *GWAS,* (2019), `https://ghr.nlm.nih.gov/primer/genomicresearch/gwastudies` Online; accessed December, 2019.

[39] A. Rauch, Z. Kutalik, P. Descombes, T. Cai, J. Di Iulio, T. Mueller, M. Bochud, M. Battegay, E. Bernasconi, J. Borovicka, *et al.*, *Genetic variation in il28b is associated with chronic hepatitis c and treatment failure: a genome-wide association study,* Gastroenterology **138**, 1338 (2010).

[40] F. Collins, *The language of life: DNA and the revolution in personalised medicine* (Profile Books, 2010).

[41] M. Alomari, F. H. Mohamed, A. W. Basit, and S. Gaisford, *Personalised dosing: printing a dose of one's own medicine,* International journal of pharmaceutics **494**, 568 (2015).

[42] 23andMe, *23andme,* (2018), https://www.23andme.com/en-eu/ Online; accessed January, 2018.

[43] E. Mullin, *It's no surprise that 23andme created a drug from customers' genetic data,* (2020), https://onezero.medium.com/its-no-surprise-that-23andme-created-a-drug-from-customers-genetic-data-1 Online; accessed January, 2020.

[44] L. Ohno-Machado, V. Bafna, A. A. Boxwala, B. E. Chapman, W. W. Chapman, K. Chaudhuri, M. E. Day, C. Farcas, N. D. Heintzman, X. Jiang, *et al.*, *idash: integrating data for analysis, anonymization, and sharing,* Journal of the American Medical Informatics Association **19**, 196 (2011).

[45] Helix, *Helix,* (2018), https://www.helix.com Online; accessed January, 2018.

[46] MyHeritage, *Myheritage,* (2018), https://www.myheritage.nl Online; accessed November, 2018.

[47] M. Fox, *Drug giant glaxo teams up with dna testing company 23andme,* (July, 2018), https://www.nbcnews.com/health/health-news/drug-giant-glaxo-teams-dna-testing-company-23andme-n894531 Online; accessed November, 2018.

[48] S. Prashad and S. Srikanthan, *23andme: Building a genetically-sound company,* (April, 2018), https://iveybusinessreview.ca/6346/23andme-building-genetically-sound-company/ Online; accessed November, 2018.

[49] Owler, *23andmeś competitors, revenue, number of employees, funding and acquisitions,* (2019), https://www.owler.com/company/23andme Online; accessed December, 2019.

[50] F. S. Collins, L. D. Brooks, and A. Chakravarti, *A dna polymorphism discovery resource for research on human genetic variation,* Genome research **8**, 1229 (1998).

[51] NLM, *Single nucleotide polymorphism,* (2018), https://ghr.nlm.nih.gov/primer/genomicresearch/snp Online; accessed January, 2018.

[52] P. M. Visscher, M. A. Brown, M. I. McCarthy, and J. Yang, *Five years of GWAS discovery,* The American Journal of Human Genetics **90**, 7 (2012).

[53] NHGRI, *Privacy in genomics,* (2019), https://www.genome.gov/about-genomics/policy-issues/Privacy Online; accessed December, 2019.

[54] B. Claerhout and G. DeMoor, *Privacy protection for clinical and genomic data: The use of privacy-enhancing techniques in medicine,* International Journal of Medical Informatics **74**, 257 (2005).

[55] D. Lazarus, *A tough lesson on medical privacy: Pakistani transcriber threatens ucsf over back pay,* San Francisco Chronicle Wednesday (2003).

[56] J. E. Lunshof, R. Chadwick, D. B. Vorhaus, and G. M. Church, *From genetic privacy to open consent,* Nature Reviews Genetics **9**, 406 (2008).

[57] B. Malin and L. Sweeney, *How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems,* Journal of biomedical informatics **37**, 179 (2004).

[58] E. Ayday, E. De Cristofaro, J.-P. Hubaux, and G. Tsudik, *Whole genome sequencing: Revolutionary medicine or privacy nightmare?* Computer **48**, 58 (2015).

[59] C. Heeney, N. Hawkins, J. de Vries, P. Boddington, and J. Kaye, *Assessing the privacy risks of data sharing in genomics,* Public health genomics **14**, 17 (2011).

[60] M. Naveed, E. Ayday, E. W. Clayton, J. Fellay, C. A. Gunter, J.-P. Hubaux, B. A. Malin, and X. Wang, *Privacy in the genomic era,* ACM Computing Surveys (CSUR) **48**, 6 (2015).

[61] X. Zhou, B. Peng, Y. F. Li, Y. Chen, H. Tang, and X. Wang, *To release or not to release: evaluating information leaks in aggregate human-genome data,* in *European Symposium on Research in Computer Security* (Springer, 2011) pp. 607–627.

[62] M. Gymrek, A. L. McGuire, D. Golan, E. Halperin, and Y. Erlich, *Identifying personal genomes by surname inference,* Science **339**, 321 (2013).

[63] B. M. Knoppers and M. Saginur, *The babel of genetic data terminology,* Nature biotechnology **23**, 925 (2005).

[64] K. L. Hudson, M. Holohan, and F. S. Collins, *Keeping pace with the times—the genetic information nondiscrimination act of 2008,* New England Journal of Medicine **358**, 2661 (2008).

[65] P. Voigt and A. Von dem Bussche, *The eu general data protection regulation (gdpr),* A Practical Guide, 1st Ed., Cham: Springer International Publishing (2017).

[66] H. G. Skinner, L. Calancie, M. B. Vu, B. Garcia, M. DeMarco, C. Patterson, A. Ammerman, and J. C. Schisler, *Using community-based participatory research principles to develop more understandable recruitment and informed consent documents in genomic research,* PLoS One **10**, e0125466 (2015).

[67] NHGRI, *Genetic discrimination,* (2019), `https://www.genome.gov/about-genomics/policy-issues/Genetic-Discrimination` Online; accessed December, 2019.

[68] R. C. Green, D. Lautenbach, and A. L. McGuire, *Gina, genetic discrimination, and genomic medicine,* New England Journal of Medicine **372**, 397 (2015).

[69] S. D. Constable, Y. Tang, S. Wang, X. Jiang, and S. Chapin, *Privacy-preserving GWAS analysis on federated genomic datasets,* BMC medical informatics and decision making **15**, S2 (2015).

[70] M. Namazi, J. R. Troncoso-Pastoriza, and F. Pérez-González, *Dynamic privacy-preserving genomic susceptibility testing,* in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security* (ACM, 2016) pp. 45–50.

[71] G. Danezis and E. De Cristofaro, *Simpler protocols for privacy-preserving disease susceptibility testing,* in *14th Privacy Enhancing Technologies Symposium, Workshop on Genome Privacy (GenoPri14), Amsterdam* (2014).

2

GENOME DATA PROCESSING AND PRIVACY TECHNIQUES

Genome data processing constitutes the use of genome data as input to an algorithm in order to obtain a desired output. This Chapter throws more light on genome data processing techniques and common algorithms that are utilized in genome data processing as well as the associated privacy preserving techniques that can be deployed in protecting the privacy and security of assets identified. We introduce the idea of genome wide association study (GWAS) and the machine learning algorithms common to this branch of genome data processing. We introduce the concept of post-GWAS, and also present algorithms that fit into this classification. Typical scenarios that are common in literature are introduced to provide a high level overview of the problem to which we propose our solutions.

2.1. TECHNIQUES IN GENOME DATA PROCESSING

2.1.1. DATA COLLECTION AND STORAGE

After a biological sample is sequenced, the data is stored in its electronic format for further processing and analysis by bioinformaticians, biostatisticians, etc. We can categorise the primary aim of genome sequencing into two steps:

- **Step 1:** For the purpose of advancement of genome related research, that may lead to discovery of previously unknown associations.
- **Step 2:** The use of the knowledge gained in the above step to answer specific genome related questions.

Although individuals and laboratories sequence genome data and retain these data, the biggest reservoir of genome data are institutions like National Center for Biotechnolgoy Information (NCBI), The European Bioinformatics Institute (EBI), DNA Data Bank of Japan (DDBJ) and China National GeneBank (CNGB) [1]. And these institutions mostly utilize their huge biorepositories for the purposes of **Step 1** above [2–5]. These institutions serve as a biorepository to genome related dataset such as DNAs, genes, single nucleotide polymorphism (SNPs) etc. They allow for submission of new data to improve the quality of their dataset, while granting access to researcher to utilize their data in scientific research. For instance, EBI states that they already have up to 273 petabytes of raw data. It is the sheer amount of data and resources at the disposal of these institutions that make it possible for them to engage in cutting edge research and discoveries, that would otherwise be impossible for a small laboratory. In fact, one of the common analysis conducted on these trove of data, is that of discovering genotype phenotype associations, using statistical algorithms and machine learning algorithms.

Individuals and smaller laboratories are more disposed pursue the goal in **Step 2**. Here, an individual might seek to use existing knowledge from genome related research, to determine the paternity of a baby, or perhaps investigate his ancestry. Observe that the success of this step is dependent on possibility and advancements made possible by **Step 1**, and data obtained in this step can always be used to improve the reliability of **Step 1**. We can conclude that both steps are in cycle that continues to improve the general understanding of the genome via genome data processing.

2.1.2. GWAS ANALYSIS

A genome-wide association study (GWAS) is a study that seeks to identify an association between a phenotype/trait and the variant of a gene locus [6–9]. Usually, these association could be that between a genetic risk factor and a complex diseases such as schizophrenia or for even a rare Mendelian diseases such as sickle cell anemia [9]. Consider a setup where a *Researcher* wishes to investigate the relationship between a disease and a some gene loci in hope of establishing an association. The *Researcher* would utilize GWAS in other to identify possible associations between the disease and a subset of the loci, thereafter further biological experiments can be directed towards the candidate loci in hopes of confirming the GWAS result. Interestingly, GWAS has been used in the field of pharmacogenetics to identify associations between DNA sequence variations and drug metabolism, efficacy, and adverse effects [9]. GWAS commonly require the collection of

cases and *control* groups to represent the group that exhibit the phenotype and another group that does not, respectively. In conducting association tests between gene locus, the statistical analysis can take the form of a quantitative traits analysis or dichotomous trait analysis [9, 10]. The quantitative traits analysis depends on statistical functions such as linear regression and *p-value* tests, while the case/control analysis can be done using logistic regression, contingency table, deep learning [11–14].

CHI-SQUARED TEST

This is a test used for determining a statistical significance margin between the expected frequency and the observed frequency of a particular study, with the use of a contingency table. For a particular SNP of interest which has two alleles, a recessive allele B and a dominant allele A . In order to test for association between a disease $\mathscr{D}$ and the SNP. A healthy group and a disease carrying group of participants are genotyped to obtain their genotype at the SNP locus. This data is used to compute a contingency table like the one in Table 2.1, where r_i , s_i , and, n_i represent genotype counts [15].

Table 2.1: A 2 x 3 genotype contingency table

	AA	AB	BB	Total
Cases	r_0	r_1	r_2	R
Controls	s_0	s_1	s_2	S
Total	n_0	n_1	n_2	N

Table 2.2: Observed allele counts

	A	B	Total
Cases	$2r_0 + r_1$	$r_1 + 2r_2$	$2R$
Controls	$2s_0 + s_1$	$s_1 + 2s_2$	$2S$
Total	$2n_0 + n_1$	$n_1 + 2n_2$	$2N$

Table 2.3: Expected allele counts

	A	B
Cases	$2R(2n_0 + n_1)/(2N)$	$2R(n_1 + 2n_2)/(2N)$
Controls	$2S(2n_0 + n_1)/(2N)$	$2S(n_1 + 2n_2)/(2N)$

To compute the observed allele counts for A and B , we generate a table similar to Table 2.2. However, the expected allele counts for the genotype is represented in Table 2.3

The Chi-square test (χ^2) is computed as:

$$\chi^2 = \sum_{i=0}^{m} \frac{(Observed - Expected)^2}{Expected} \cdot \tag{2.1}$$

LINEAR REGRESSION

When the test of association is between a genotype and a continuous-valued phenotype, generalized linear models such as linear regression technique is adopted [9, 12]. A typical logistic regression equation takes the form:

$$y = \sum_{i=1}^{k} x_i \beta_i + \beta_0 + \epsilon \,. \tag{2.2}$$

From Eq. 2.1, k is the number of genotype loci considered, x_i values are the values of the genotypes, β_i values represent the coefficients of the corresponding genotypes also known as model parameters, while β_0 is the intercept on y-axis, and ϵ is the error variable. Where $k = 1$, which means only one genotype is analyzed, we have a univariate regression model and where $k > 1$, it is considered a multivariate regression model with more than one genotype simultaneously analyzed [15, 16]. In [12], Sikorska et. al demonstrate how other non-genotype covariates such as age, weight and height can be combined alongside a genotype variable in order to predict a dependent variable y , as shown in Eq. 2.2

LOGISTIC REGRESSION

Logistic regression is closely related to its linear counterpart but is rather used for dichotomous outcomes unlike the linear model preferred for continuous-valued outcomes. Let the response variable y represent the disease $\mathscr{D}$ in an individual. For individual i , where $(1 \leq i \leq N)$, and genotype loci j , where $(1 \leq j \leq k)$, we have:

$$y = \begin{cases} 0, & \text{if } \mathscr{D} = 1 \\ 1, & \text{otherwise} \cdot \end{cases} \tag{2.3}$$

$$x_{i,j} = \begin{cases} 0, & \text{if genotype}_j = AA \\ 1, & \text{if genotype}_j = AB \\ 2, & \text{if genotype}_j = BB \cdot \end{cases} \tag{2.4}$$

For a logistic regression model that predicts the value of trait y given the values for the genotypes x_is, we proceed as follows. Let $p_i = Pr(y_i | x_1, \ldots, x_j)$ represent the probability of the trait being y_i for the genotypes $(x_1, \ldots, x_j)$. We define the logit function as:

$$logit(p_i) \;=\; \log_e \left[\frac{p_i}{1 - p_i} \right] = \sum_{j=1}^{k} x_j \beta_j + \beta_0 + \epsilon \cdot \tag{2.5}$$

$$p_i = \frac{1}{1 + e^{-(\sum_{j=1}^{k} x_j \beta_j + \beta_0 + \epsilon)}} \cdot \tag{2.6}$$

The logistic model is common in medical epidemiology due to its ability to clearly classify the occurrence of an event using the sigmoid function as presented in Eq. 2.6 [12, 16].

Both the linear regression model and the logistic variant offer the advantages of parallelization in some part, due to their linear combination properties [12, 17]. Even though these analyses assume that the data sources x_i values originate from a single database or institution, the converse is equally obtainable in the wild. In most cases, more than one institution will own and house the data, and they often seek for efficient and protected means for data sharing in a collaborative setting.

2.1.3. POST-GWAS ANALYSIS

Once information is extracted from the genome during the GWAS phase, these information is now data to be utilized in the Post-GWAS phase for further computations such as predicting trait susceptibility [18, 19]. This phase of post-GWAS analysis can be viewed as the prediction phase of a machine learning process [20]. Data generated for the models during the GWAS phase can then be put to use by predicting status of previously unclassified individuals [21]. Other applications are paternity testing, diagnosis of diseases, and forensic evidence for solving criminal cases.

2.2. PRIVACY ENHANCING TECHNOLOGIES

Due to the privacy concerns listed in Section 1.4 and the analyses of data described in 2.1.2 the need for efficient privacy-preserving techniques that are applicable to genome dataset becomes obvious. The research community as well as industry players look for methods for providing privacy-guarantees in the face of assumed privacy-threats. Even though there are multiple techniques that can provide privacy guarantees independently, they are often accompanied by costs. These costs can assume the form of computational overhead due to the complexity of the implementation, storage overhead as a result of data expansion, and communication overhead by virtue of multiple rounds of communications. The choice of a suitable privacy enhancing technique is usually determined by the trade-off between utility versus privacy that is obtainable. In this section, we briefly introduce some of these privacy-preserving techniques utilized for the protection of sensitive data. We first introduce the concept of security model as will be utilized by the various settings where the privacy enhancing technologies are adopted.

2.2.1. SECURITY MODEL

In order to process genome data in any useful scenario, data is often required to be transmitted from one party to another. In some case, data transfer could be just for the sake of storage, while in other scenarios it might be for analysis. Because of this, it becomes pertinent to clearly define the roles of each party, as well as the security assumptions that are acceptable in any such scenario, which includes the capability and behaviour of the adversaries [22]. This is encapsulated in the concept of security model. There are two major types of adversaries [23]:

- **Semi-honest adversary:** This adversarial model assumes that all parties defined in a protocol will follow the specifications of the protocol and never deviate from it. This can be considered as a passive adversary who does not intentionally act maliciously during the protocol execution, but can observe transcripts of the protocol in order to deduce private information from them [23–25]. In this model, there

is some degree of trust amongst the parties hence the name *honest-but-curious* adversary. This adversary is weak and usually does not require extreme privacy measures to guarantee privacy.

- **Malicious adversary:** Contrary to the weak adversary described as *honest-but-curious*, the malicious setting defines a very strong adversary who can decide to deviate extensively from the protocol description [23–26]. In this model, there is no trust whatsoever amongst the parties, and every operation would require a verification mechanism before trusting the output. This is an active adversary and more difficult to protect against without degrading the efficiency of the protocol.

Other than the two major models listed above, an intermediary adversary known as a *covert adversary* is also possible [23, 27]. The *covert* adversary lies in between the semi-honest and the malicious adversaries. In this model, the adversary is allowed to deviate from the protocol. However, the malicious will be caught cheating by the honest parties with some define probability. This is an active adversary operating in an high risk environment with a substantial probability of being caught.

2.2.2. MASKING

Masking is a cryptographic technique utilized to obfuscate the value of a secret, such that when the masked value is viewed by an adversary, it obfuscates the original secret [25, 28, 29]. This method offers statistical security and can be achieved either by *additive masking* or *multiplicative masking*. Given a secret value m , and a random value r , additive and multiplicative masking can be achieved as follows:

$$m_s = m + r \mod N, \tag{2.7}$$

$$m_s = m \cdot r \mod N. \tag{2.8}$$

Data masking is commonly adopted for data aggregation protocols, where little distortions in the value of the secret helps in obfuscation while the aggregate value can still be obtained by subtracting the aggregate noise. However, where more complex operations other than addition is required, masking do not offer a lot towards the utility of the data.

2.2.3. SECRET SHARING

This is a privacy enhancing technique that provides theoretical security by distributing the information amongst various parties whom are considered to be non-collaborating. The non-collusion assumption is relevant to the security of this scheme, since parties could simply collude to recover some part or whole of the secret [30]. In such schemes, only a predefined number of the set can come together to recover the secret, and it is commonly denoted as t-out-of-n schemes. Where some well known n secret sharing schemes include the Shamir's secret sharing [31] and the multi-linear secret sharing scheme [32, 33]. Although this scheme is usually fast and easy to implement, the drawbacks usually include finding multiple parties that are non-colluding, and the communication overhead associated with evaluating any meaningful function.

2.2.4. DIFFERENTIAL PRIVACY

This technique offers statistical protection to data by minimizing the amount of information made public, hence its name, *statistical disclosure control* [34, 35]. Although this is a commonly applied technique for data protection [36], it does not provide the provable security/privacy guarantees that are obtainable in some other privacy enhancing techniques. Also, applying differential privacy to a dataset adds noise to the data and could constrain the utility of such data [34, 37].

2.2.5. HOMOMORPHIC ENCRYPTION

A homomorphic encryption (HE) scheme allows for arbitrary algebraic operations to be performed on ciphertexts. Let $Enc_{pk}(\cdot)$ and $Dec_{sk}(\cdot)$ represent encryption and decryption functions respectively. (m_1 , m_2) are two messages and k is a scalar value, while $\boxplus$, $\boxdot$ and $\boxtimes$ are arbitrary operations on the ciphertexts. Then, homomorphism is defined as follows:

$$Dec_{sk}(Enc_{pk}(m_1) \boxplus Enc_{pk}(m_2)) = m_1 + m_2 \,, \tag{2.9}$$

$$Dec_{sk}(Enc_{pk}(m_1) \boxdot Enc_{pk}(m_2)) = m_1 \cdot m_2 \,, \tag{2.10}$$

$$Dec_{sk}(Enc_{pk}(m_1) \boxtimes k) = m_1 \cdot k \,. \tag{2.11}$$

HE schemes can further be classified into two types:

- **Partial homomorphic schemes:** In this class of homomorphic encryption scheme, only one of *addition* or *multiplication* operation is obtainable [38]. These types of schemes are usually limited in operations and can be combine with other schemes in an interactive setting in other to achieve the properties they lack. Examples of these schemes include Paillier homomorphic scheme [39] which offers only addition properties, the additively homomorphic scheme by Peters et. al [40], and the DGK scheme [41]. Other multiplicatively homomorphic schemes include the ElGamal scheme [42] and the commonly used RSA scheme [43].

- **Fully homomorphic schemes:** Fully HE schemes allow for addition and multiplication operations to be executed on ciphertexts. The properties of fully homomorphic schemes make them much suitable for evaluating functions in an ideal environment, unlike the partial homomorphic schemes [38, 44]. Despite the advantages that accompany the fully homomorphic schemes, their major drawback comes in the form of computational overhead. Due to the complexity of realizing a fully homomorphic evaluation, the computation cost and the data expansion properties make them inefficient for deployment in the wild when compared to other privacy enhancing techniques. Craig Gentry was the first to realise a fully homomorphic scheme with the proposal in [45], thereafter, other fully homomorphic schemes [38, 46–48] have been proposed. Most fully homomorphic schemes are designed using lattice structures and their security are based on the hardness of learning with errors (LWE) problem. Fully homomorphic constructions have been considered to be quantum resistant schemes, due to the fact that no known quantum attack has been proven successful [49].

- **Levelled homomorphic schemes:** This is a variant of the fully homomorphic scheme, with some limitation on number of operations. In fact, the *levelled* homomorphic scheme, also known as somewhat homomorphic scheme is the practical variant of the fully homomorphic scheme. The idea is to limit the number multiplication operations to a predefined threshold, known as *depth*. The *depth* of a *levelled* homomorphic scheme is determined by the parameters during the setup phase of the scheme. Unlike the fully homomorphic variant, the *levelled* schemes are commonly designed by replacing the lattice structure with a large polynomial, and as a result, relying on the hardness of ring learning with errors problem (ring-LWE) for the security. *Levelled* homomorphic schemes offer better efficiency both with respect to computational and storage overheads, when compared to their fully variants. For this reasons, they have been adopted both in research community and in industry. Example of these schemes include the BGV scheme [50], the FV scheme [51], the Brakerski's scheme [52], the YASHE scheme [53], and the HEAAN scheme [54].

Even though homomorphic encryption provides provable security based on some computationally hard problem, they are usually associated with huge computational overheads when compared to other privacy enhancing technologies. The choice of implementing a homomorphic scheme depends on the specific problem scenario and how much utility can be traded for the privacy guarantees.

2.2.6. SECURE MULTI-PARTY COMPUTATION

A secure multi-party computation is an interactive cryptographic protocol that allows for two or more mistrusting parties to jointly compute a function using their private data as input [23, 55, 56]. It allows for the output of the desired function to be public but the contributed inputs remains private upon the assumption that each party does not digress from the rules of the protocol. Secure multi-party computations are commonly designed in a semi-honest security model, because the adversary is considered to be able to control some parties in the protocol. However, there have been attempts to design secure multi-party computation protocols that are secure under the malicious model [57, 58]. Depending on the problem setting, secure multi-party computation protocols can utilize any of the above mentioned privacy enhancing techniques as a sub-protocol in order to solve the problem. Examples of secure multi-party computation protocols can be seen in [58–60].

2.3. PRIVACY-PRESERVING SCENARIOS

In order to succinctly discuss the common scenarios encountered during genome data processing, we first introduce the entities or stakeholders of a typical genome ecosystem, at least, to the extent that it is discussed in this thesis.

2.3.1. STAKEHOLDERS

The entities introduced here have been well introduced and discussed in privacy-preserving-genomics related literature [61–64].

- *Patient/Data Owner:* this represents the entity/parties who owns a set of genome

data. It does not necessarily need to be a sick individual, since genome data processing can also be carried out with the genome of a healthy individual. While a set of genome data primarily belongs to a unique individual, our definition of a genome data owner is expanded to include close relatives of the original owner. This definition is valid because an individual shares some part of his genome with his parents and even sibling, making it feasible for such relationships to be identified through genome data processing. Depending on the specific scenario, the genome data can be stored on a device which is securely preserved by the Data Owner, or, it can be stored with a cloud infrastructure on behalf of the Data Owner.

Figure 2.1: A Patient or Data Owner

- *Certified Authority/Sequencer:* this entity represents a known authority, who is certified to sequence the *in vitro* sample of a genome in order to obtain the *in silico* version of the same genome. This authority may be a commercial company that provides these services for profit, but is expected to operate within the dictates of the laws and observe ethical standards.

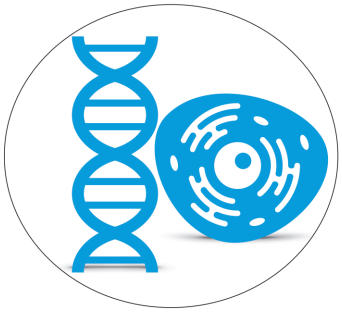

Figure 2.2: A Certified Authority or Sequencer

- *Storage and Processing Unit:* this party serves as a cloud infrastructure that offers services such as: data storage and data processing for both high-storage-demanding data and computationally intensive operations. This party can be a commercial company whose interest is in making profit. As a result of the sensitivity of the

data, this party is expected to provide protection for the data resident with it, where such data is stored non-encrypted. And in cases where the genome data is encrypted before storing with this party, it is expected that operations required for processing the genome returns results that are consistent with the actual computations. The storage and processing unit is assumed to own the software and hardware necessary to manage the data, and always has sufficient resources to perform his functions.

Figure 2.3: Storage and Processing Unit

- *Medical Personnel:* this party constitutes hospitals and medical staff who might be interested in recommending treatment for a patient or offering other related services to data owners. Although they do not own the data nor store them, the rest of the parties rely on the expertise of the Medical Personnel when deciding which medical tests to undergo and also for the interpretation of results.

Figure 2.4: Storage and Processing Unit

- *Researcher:* this party describes the role of a research institution or that of a pharmaceutical institution where the interest would include: studying the genome, identifying new genes, investigating gene-phenotype relations and many more. While these parties may not own the genome, they are interested in using clusters of genome data in available data banks in order to aid their research and new discoveries.

Figure 2.5: A Researcher

2.3.2. SCENARIOS

We present some common scenarios where demands for privacy-preserving genome data processing has been rife.

- **Scenario 1:** How can a *patient* privately compute a diseases susceptibility test without leaking sensitive data to other participating entities?

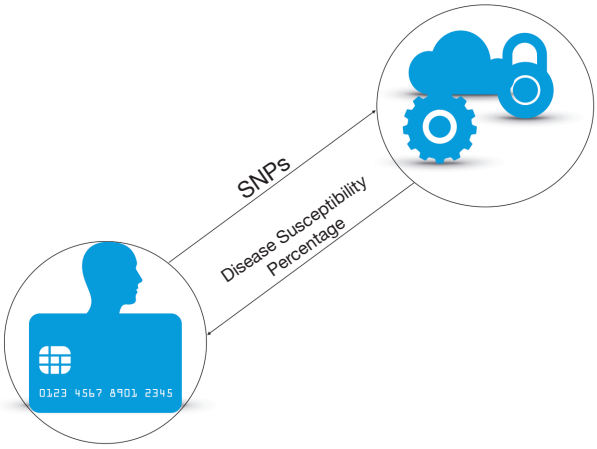

Figure 2.6: Scenario One: A patient utilizing the DST services with private genome data

This is a commonly described problem scenario within the privacy-preserving genome data processing community, hence the various solutions that have previously been contributed towards it. Most notable is the work by Ayday et al. [62] where the first attempt was made to provide privacy for this scenario. The solution by Ayday adopted homomorphic encryption and multi-party computation as the privacy enhancing techniques for solving this problem. While their solution provides privacy for the sensitive data, it accumulates computational and storage costs due to the data expansion and computational complexity properties associated with homomorphic encryption. Other attempts to propose solutions to this problem can

be seen in [18, 63, 64]. In this thesis, we seek to develop novel protocols that do not suffer from the inefficiency nor privacy gaps observed in existing proposals.

- **Scenario 2:** How can a *storage and processing unit* perform simple statistical computations such as chi-square analysis without learning privacy-invasive information from the genome?

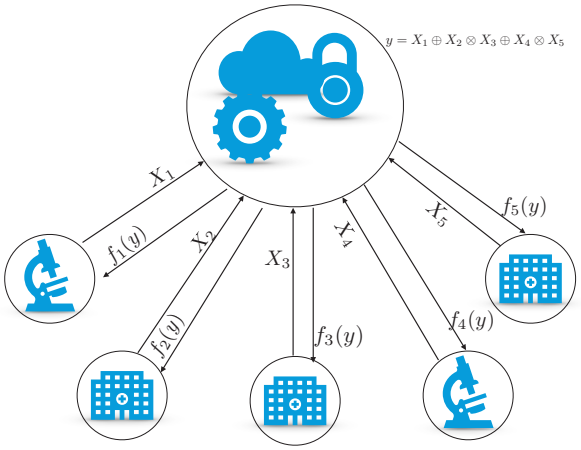

Figure 2.7: Scenario Two: A Processing Unit that receives protected data from multiple sources, and computes various functions for users

How can an authenticated researcher privately analyse a large data bank to learn information that are not privacy invasive or identifiable to data owners. Such information could start from running simple statistical functions on the rich dataset, to executing complex machine learning algorithms on the dataset. Similar description of this problems can be found in [61, 65–67], where the authors deployed various techniques including homomorphic encryption, multi-party computation, and secret sharing techniques.

- **Scenario 3:** How can multiple *storage and processing unit*s collaborate to analyse their data in hope of generating a public model using their privately contributed data?

  The aim here is to introduce a machine learning environment that allows for collaboration within multiple data owners, in order to further enrich the quality of the generated model while preserving the privacy each party's genome dataset. At the time of writing this thesis, this problem as described in Fig. 2.8 was still an open problem due to its lack of an efficient privacy-preserving solution. This scenario was one of the challenges proposed by iDASH in the 2019 competition, and our proposal was a novel and successful solution at the competition.

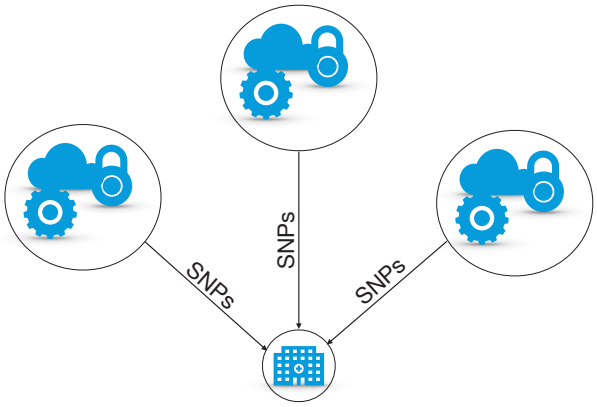

Figure 2.8: Scenario Three: A collaborative machine learning model generation between mistrusting parties

REFERENCES

[1] J. Pevsner, *Bioinformatics and functional genomics* (John Wiley & Sons, 2015).

[2] EBI, *The european bioinformatics institute,* (2019), https://www.ebi.ac.uk Online; accessed January, 2020.

[3] NCBI, *The national center for biotechnology information,* (2019), https://www.ncbi.nlm.nih.gov Online; accessed January, 2020.

[4] CNGB, *China national genebank,* (2019), https://en.genomics.cn/en-gene.html Online; accessed January, 2020.

[5] D. D. B. of Japan, *Bioinformation and ddbj center,* (2019), https://www.ddbj.nig.ac.jp/index-e.html Online; accessed January, 2020.

[6] A. Korte and A. Farlow, *The advantages and limitations of trait analysis with GWAS: a review,* Plant methods **9**, 29 (2013).

[7] S. B. Zaghlool, B. Kühnel, M. A. Elhadad, S. Kader, A. Halama, G. Thareja, R. Engelke, H. Sarwath, E. K. Al-Dous, Y. A. Mohamoud, *et al.*, *Epigenetics meets proteomics in an epigenome-wide association study with circulating blood plasma protein traits,* Nature Communications **11**, 1 (2020).

[8] NLM, *GWAS,* (2019), https://ghr.nlm.nih.gov/primer/genomicresearch/gwastudies Online; accessed December, 2019.

[9] W. S. Bush and J. H. Moore, *Genome-wide association studies,* PLoS computational biology **8**, e1002822 (2012).

[10] J. Huang and Y. Jiang, *Genetic linkage analysis of a dichotomous trait incorporating a tightly linked quantitative trait in affected sib pairs,* The American Journal of Human Genetics **72**, 949 (2003).

[11] J. Stankovich, *Statistical analysis of genome-wide association (GWAS) data,* http://bioinformatics.org.au/ws09/presentations/Day3_JStankovich.pdf Online; accessed November, 2018.

[12] K. Sikorska, E. Lesaffre, P. F. Groenen, and P. H. Eilers, *GWAS on your notebook: fast semi-parallel linear and logistic regression for genome-wide association studies,* BMC bioinformatics **14**, 166 (2013).

[13] Y. Wang, J. G. Klijn, Y. Zhang, A. M. Sieuwerts, M. P. Look, F. Yang, D. Talantov, M. Timmermans, M. E. Meijer-van Gelder, J. Yu, *et al.*, *Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer,* The Lancet **365**, 671 (2005).

[14] H. Xie, J. Li, Q. Zhang, and Y. Wang, *Comparison among dimensionality reduction techniques based on random projection for cancer classification,* Computational biology and chemistry **65**, 165 (2016).

[15] C. A. de Leeuw, J. M. Mooij, T. Heskes, and D. Posthuma, *Magma: generalized gene-set analysis of GWAS data,* PLoS computational biology **11**, e1004219 (2015).

[16] P. F. O'Reilly, C. J. Hoggart, Y. Pomyen, F. C. Calboli, P. Elliott, M.-R. Jarvelin, and L. J. Coin, *Multiphen: joint model of multiple phenotypes can increase discovery in GWAS,* PloS one **7**, e34861 (2012).

[17] A. A. Shabalin, *Matrix eqtl: ultra fast eqtl analysis via large matrix operations,* Bioinformatics **28**, 1353 (2012).

[18] E. Ayday, J. L. Raisaro, and J.-P. Hubaux, *Personal use of the genomic data: Privacy vs. storage cost,* in *2013 IEEE Global Communications Conference (GLOBECOM)* (IEEE, 2013) pp. 2723–2729.

[19] D. L. Armstrong, R. Zidovetzki, M. E. Alarcón-Riquelme, B. P. Tsao, L. A. Criswell, R. P. Kimberly, J. B. Harley, K. L. Sivils, T. J. Vyse, P. M. Gaffney, *et al.*, *GWAS identifies novel sle susceptibility genes and explains the association of the hla region,* Genes and immunity **15**, 347 (2014).

[20] M. W. Libbrecht and W. S. Noble, *Machine learning applications in genetics and genomics,* Nature Reviews Genetics **16**, 321 (2015).

[21] Y. Smith, *Applications of genomics,* (2019), https://www.news-medical.net/life-sciences/Applications-of-Genomics.aspx Online; accessed January, 2020.

[22] Z. Erkin and G. Tsudik, *Private computation of spatial and temporal power consumption with smart meters,* in *International Conference on Applied Cryptography and Network Security* (Springer, 2012) pp. 561–577.

[23] C. Hazay and Y. Lindell, *Efficient secure two-party protocols: Techniques and constructions* (Springer Science & Business Media, 2010).

[24] Y. Lindell and B. Pinkas, *Privacy preserving data mining,* in *Annual International Cryptology Conference* (Springer, 2000) pp. 36–54.

[25] F. Knirsch, G. Eibl, and D. Engel, *Error-resilient masking approaches for privacy preserving data aggregation,* IEEE Transactions on Smart Grid **9**, 3351 (2016).

[26] J. Brickell and V. Shmatikov, *Privacy-preserving graph algorithms in the semi-honest model,* in *International Conference on the Theory and Application of Cryptology and Information Security* (Springer, 2005) pp. 236–252.

[27] Y. Aumann and Y. Lindell, *Security against covert adversaries: Efficient protocols for realistic adversaries,* in *Theory of Cryptography Conference* (Springer, 2007) pp. 137–156.

[28] C. Castelluccia, A. C. Chan, E. Mykletun, and G. Tsudik, *Efficient and provably secure aggregation of encrypted data in wireless sensor networks,* ACM Transactions on Sensor Networks (TOSN) **5**, 20 (2009).

[29] E. Ayday, J. L. Raisaro, U. Hengartner, A. Molyneaux, and J.-P. Hubaux, *Privacy-preserving processing of raw genomic data,* in *Data Privacy Management and Autonomous Spontaneous Security* (Springer, 2013) pp. 133–147.

[30] A. Beimel, *Secret-sharing schemes: a survey,* in *International Conference on Coding and Cryptology* (Springer, 2011) pp. 11–46.

[31] A. Shamir, *How to share a secret,* Communications of the ACM **22**, 612 (1979).

[32] M. Bertilsson and I. Ingemarsson, *A construction of practical secret sharing schemes using linear block codes,* in *International Workshop on the Theory and Application of Cryptographic Techniques* (Springer, 1992) pp. 67–79.

[33] M. van Dijk, *A linear construction of perfect secret sharing schemes,* in *Workshop on the Theory and Application of of Cryptographic Techniques* (Springer, 1994) pp. 23–34.

[34] C. Dwork, A. Roth, *et al.*, *The algorithmic foundations of differential privacy,* Foundations and Trends® in Theoretical Computer Science **9**, 211 (2014).

[35] C. Dwork, *Differential privacy: A survey of results,* in *International conference on theory and applications of models of computation* (Springer, 2008) pp. 1–19.

[36] 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* (2015) pp. 1286–1297.

[37] S. Simmons and B. Berger, *Realizing privacy preserving genome-wide association studies,* Bioinformatics **32**, 1293 (2016).

[38] P. Martins, L. Sousa, and A. Mariano, *A survey on fully homomorphic encryption: An engineering perspective,* ACM Computing Surveys (CSUR) **50**, 1 (2017).

[39] P. Paillier, *Public-key cryptosystems based on composite degree residuosity classes,* in *Advances in cryptology?EUROCRYPT?99* (Springer, 1999) pp. 223–238.

[40] A. Peter, M. Kronberg, W. Trei, and S. Katzenbeisser, *Additively homomorphic encryption with a double decryption mechanism, revisited,* in *International Conference on Information Security* (Springer, 2012) pp. 242–257.

[41] I. Damgard, M. Geisler, and M. Kroigard, *Homomorphic encryption and secure comparison,* International Journal of Applied Cryptography **1**, 22 (2008).

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

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

[44] F. Armknecht, C. Boyd, C. Carr, K. Gjøsteen, A. Jäschke, C. A. Reuter, and M. Strand, *A guide to fully homomorphic encryption.* IACR Cryptology ePrint Archive **2015**, 1192 (2015).

[45] C. Gentry, *Fully homomorphic encryption using ideal lattices,* in *Proceedings of the forty-first annual ACM symposium on Theory of computing* (2009) pp. 169–178.

[46] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, *Fully homomorphic encryption over the integers,* in *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, 2010) pp. 24–43.

[47] N. P. Smart and F. Vercauteren, *Fully homomorphic encryption with relatively small key and ciphertext sizes,* in *International Workshop on Public Key Cryptography* (Springer, 2010) pp. 420–443.

[48] Z. Brakerski and V. Vaikuntanathan, *Fully homomorphic encryption from ring-lwe and security for key dependent messages,* in *Annual cryptology conference* (Springer, 2011) pp. 505–524.

[49] A. Martin, C. Melissa, C. Hao, D. Jintai, G. Shafi, G. Sergey, H. Shai, H. Jeffrey, L. Kim, L. Kristin, L. Satya, M. Daniele, M. Dustin, M. Travis, S. Amit, and V. Vinod, *Homomorphic encryption standard,* (2018), `http://homomorphicencryption.org/wp-content/uploads/2018/11/HomomorphicEncryptionStandardv1.1.pdf` Online; accessed January, 2020.

[50] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, *(leveled) fully homomorphic encryption without bootstrapping,* ACM Transactions on Computation Theory (TOCT) **6**, 1 (2014).

[51] J. Fan and F. Vercauteren, *Somewhat practical fully homomorphic encryption.* IACR Cryptology ePrint Archive **2012**, 144 (2012).

[52] Z. Brakerski, *Fully homomorphic encryption without modulus switching from classical gapsvp,* in *Annual Cryptology Conference* (Springer, 2012) pp. 868–886.

[53] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig, *Improved security for a ring-based fully homomorphic encryption scheme,* in *IMA International Conference on Cryptography and Coding* (Springer, 2013) pp. 45–64.

[54] J. H. Cheon, A. Kim, M. Kim, and Y. Song, *Homomorphic encryption for arithmetic of approximate numbers,* in *International Conference on the Theory and Application of Cryptology and Information Security* (Springer, 2017) pp. 409–437.

[55] O. Goldreich, *Secure multi-party computation,* Manuscript. Preliminary version , 86 (1998).

[56] R. Cramer, I. B. Damgård, *et al.*, *Secure multiparty computation* (Cambridge University Press, 2015).

[57] Y. Lindell and B. Pinkas, *An efficient protocol for secure two-party computation in the presence of malicious adversaries,* in *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, 2007) pp. 52–78.

[58] P. Mohassel and M. Franklin, *Efficiency tradeoffs for malicious two-party computation,* in *International Workshop on Public Key Cryptography* (Springer, 2006) pp. 458–473.

[59] T. Schneider and M. Zohner, *GMW vs. Yao? efficient secure two-party computation with low depth circuits,* in *International Conference on Financial Cryptography and Data Security* (Springer, 2013) pp. 275–292.

[60] Y. Huang, D. Evans, J. Katz, and L. Malka, *Faster secure two-party computation using garbled circuits.* in *USENIX Security Symposium*, Vol. 201 (2011) pp. 331–335.

[61] K. Lauter, A. López-Alt, and M. Naehrig, *Private computation on encrypted genomic data,* in *Progress in Cryptology-LATINCRYPT 2014* (Springer, 2014) pp. 3–27.

[62] 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* (ACM, 2013) pp. 95–106.

[63] G. Danezis and E. De Cristofaro, *Simpler protocols for privacy-preserving disease susceptibility testing,* in *14th Privacy Enhancing Technologies Symposium, Workshop on Genome Privacy (GenoPri14), Amsterdam* (2014).

[64] M. Namazi, J. R. Troncoso-Pastoriza, and F. Pérez-González, *Dynamic privacy-preserving genomic susceptibility testing,* in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security* (ACM, 2016) pp. 45–50.

[65] Y. Zhang, W. Dai, X. Jiang, H. Xiong, and S. Wang, *Foresee: Fully outsourced secure genome study based on homomorphic encryption,* in *BMC medical informatics and decision making*, Vol. 15 (Springer, 2015) p. S5.

[66] S. E. Fienberg, A. Slavkovic, and C. Uhler, *Privacy preserving GWAS data sharing,* in *2011 IEEE 11th International Conference on Data Mining Workshops* (IEEE, 2011) pp. 628–635.

[67] L. Kamm, D. Bogdanov, S. Laur, and J. Vilo, *A new way to protect privacy in large-scale genome-wide association studies,* Bioinformatics **29**, 886 (2013).

3

PRIVACY-PRESERVING GWAS

In the search for a holistic privacy-preserving approach for protecting sensitive data such as the genome, one must not fail to consider the privacy of the component units that make up the whole. GWAS is only a component upon which a privacy-preserving solution is not only possible but has been demonstrated.

Parts of this chapter have been published as:
(1) Ugwuoke, C., Erkin, Z., & Lagendijk, R. L. (2017). Privacy-safe linkage analysis with homomorphic encryption. *In 2017 25th European Signal Processing Conference (EUSIPCO) (pp. 961-965). IEEE.*
(2) Ugwuoke, C., Erkin, Z., & Lagendijk, I. (2016). A Privacy-Preserving GWAS Computation with Homomorphic Encryption. *In 37th WIC Symposium on Information Theory in the Benelux/6th WIC/IEEE SP Symposium on Information Theory and Signal Processing in the Benelux, Louvain, Belgium (pp. 166-73).*

3.1. LITERATURE STUDY ON PRIVACY-PRESERVING GENOME DATA PROCESSING

The human genome is a fully detailed representation of how a single person is constructed on the molecular level. Throughout the years, researchers have been more and more capable of associating different genes in our genome to specific traits which can be disease or phenotypes. By way of doing so, scientists have been able to better understand how humans and other organisms are configured leading to new insights, primarily in the medical sector. Obtaining the genome of an organism happens via a process called DNA sequencing where the exact composition of nucleotides in the genome are retrieved. Over time, this process has become much faster and cheaper leading to better research in the field of genomics. Additionally, the emergence of a novel commercial sector based on sequencing techniques has led to companies offering digitised versions of a genome at a relatively cheap price. Since genomic data is able to provide a lot of sensitive information about an individual, efficient privacy-preserving techniques becomes an integral requirement in processing genome data. In this survey, we describe the current state of privacy-preserving techniques that have been used to protect genomic data. We performed a demarcated literature study on different scenarios where genomic data is being handled. As a result, we present a categorisation of the various scenarios based on their privacy technique, parties involved, and tests that can be performed on the genomic data. From the study we conclude that no “one size fits all" solution exists for privacy-preserving genome data processing and that, currently, many genome data processing require an adjusted version of existing techniques.

3.1.1. INTRODUCTION

In the last two decades, genome sequencing technology has experienced a rapid boost in development. Main advancements are being made at enabling faster and cheaper sequencing of a single person's genome [1]. These advancements have been a great benefit to the medical community allowing research to progress faster. The process of sequencing DNA already started in the early 70s [2]. Short after, the first full genome was sequenced in 1977 [3], making it possible to obtain the exact composition of nucleotide bases in the genome of a bacteriophage. Less than 25 years later, the Human Genome Project came along, aiming at sequencing the full genome of a human being. Ever since, many other human DNA has been sequenced and subsequently stored in a digital way [4]. Also, in the commercial sector we see companies, like 23andMe [5], profiting from the simplification of the genome sequencing process [5]. But, as with many advancements, techniques for handling genomic data were developed and implemented without primarily considering the security and privacy implications. This lack of security/privacy-by-design could result to the threat to privacy of an individual that partakes in a genome-data-dependent experiments.

Digitising human genomic information introduces new discussions in the field of ethics, law, and social sciences primarily due to the privacy concerns that come together with handling such sensitive data [6–8]. Our human genome carries a lot of personal and delicate information not only about the individual itself, but also about this individual's siblings, ancestors and other relatives. This information reveals the ethnicity of the human being as well as susceptibility to certain diseases. As our medical knowledge about the human genome will increase over the years, more information will become visible in the genomic data of an individual. Sharing this data is, hence, not only the concern of the individual from which it gets sequenced, but also of the people related to that individual [9].

The main goal of sequencing the genome is to perform research or seek answers to specific questions by computationally processing the data in order to gain these insights. These insight could vary from seeking information about an individual like in cases where a paternity or maternity test is required, to search for association between genes and traits, where genome from multiple individuals will be needed. Incentives to do so can come from the individual itself or be imposed by a doctor or governmental entity such as some court or the police. In this paper, the focus is on the former circumstance. The main problem for such an individual is the lack of resources, which can be storage capabilities, computational power, an algorithm, etcetera. These resources can then be provided by a third party, often through a cloud environment. Although cloud infrastructures offer a cost-effective solution for resource scarcity, they are often seen as an untrustworthy environment. To solve these privacy issues, solutions based on homomorphic encryption, differential privacy, and many other techniques have been proposed. Unfortunately, with these techniques, trade offs have to be made between the level of privacy, accuracy, computational overhead, and various other dimensions.

This work aims at providing an answer to the question: what is the current status quo on preservation of privacy when managing genomic data? Concretely, we look at 1) which techniques are currently used for privacy preserving genome testing, 2) on which types of genomic testing has these techniques been applied, and 3) how many actors

need to be involved once the genome has been sequenced? What is the communication overhead? We identify a number of recent studies and provide a deeper analyses on them. Based on our research we provide 14 scenarios that we can categorise in 3 different categories. As a result, we show that external parties are currently needed in the process of genomic testing to provide resources, such as computational power, missing by the individual. Furthermore, we show that most tests require an adjusted version of a privacy preserving technique and that these techniques differ and depend on the test needed to be performed.

To the best of our knowledge there are only two papers that performed a similar study on the state of genomic privacy. Ayday et al. concluded in 2015 that, while being a fruitful technique for biomedical studies, DNA sequencing still proposes some privacy challenges that need to be solved [10]. Concretely, the paper proposed some guidelines issuing the challenges that remain in genome privacy such as accessibility, accuracy of testing, and efficiency of testing. Since genomic data currently needs to be handled by different entities more parties are able to retrieve information out of the genomic data and, hence, damage the privacy of the individual to which the genome belongs. Furthermore, accuracy of performing the tests digitally and encrypted need to remain at least the same as when handling such data without encryption or *in vitro*. Also computational and communication costs should be low enough to make the techniques practical. In 2017, Bradley et al. concluded that genomic privacy still was not where it should be. [11]

The rest of the paper will be structured as follows: we will go over some preliminary privacy techniques that will be used throughout the paper. This will be followed by an illustration of the scope of this study together with a report on our literature study. This paper is concluded with the results and conclusions of our literature research.

3.1.2. PRELIMINARIES

We introduce some of the concepts and privacy-preserving techniques that are commonly identified in the protocols designed to protect genome data processing.

SINGLE NUCLEOTIDE POLYMORPHISM

The DNA is made up of nucleotide bases from the set $\{"A", "G", "T", "C"\}$, and portions of the genome are often copied or inherited by individual. However, when there are subtle changes to a section of the gene such that a single nucleotide distinguishes two or more individuals, a single nucleotide polymorphism (SNP) is said to have occurred. Single Nucleotide Polymorphisms are the most common genetic variation of the genome among the population. A SNP always denotes a the substitution, deletion, or insertion of a single nucleotide in a specific gene locus [12]. SNPs are commonly used as biological markers for individuals, and every individual is said to have as much as 3 million SNPs [12].

HOMOMORPHIC ENCRYPTION

Homomorphic encryption enables data to be processed non-interactively in the encrypted domain without having to decrypt the ciphertext. Operations performed on encrypted data yield the same result after decryption as if they were performed on the

plaintext. Common operations that are obtainable using homomorphic encryption include basic arithmetic operations like addition, subtraction, and multiplication. Some types of homomorphic encryption schemes known as partial homomorphic schemes offer only either addition or multiplication property, and example include RSA [13, 14], Paillier[15] and ElGamal[16] schemes. Other types of homomorphic encryption schemes are able to provide for both additive and multiplicative properties, and they are known as fully homomorphic schemes. Examples of fully homomorphic encryption schemes include the construction by Gentry [17], and variants such as [18–22]. Most fully homomorphic schemes and their variants leverage computationally hard lattice problems to construct the required operations, making them suitable to be considered for post-quantum cryptography [23]. Although homomorphic encryption provides data confidentiality without the need for interactive protocols, it however attracts huge computational overhead, making them not practical for most settings where resources are limited.

DIFFERENTIAL PRIVACY

Differential privacy also known as statistical disclosure control is a common privacy-preserving technique used for statistical dataset [24]. It is considered to provide extremely accurate statistical information about a database, while limiting the disclosure of information about the individual records [25]. Unlike some other techniques like homomorphic encryption, differential privacy is not as computationally intensive and could be deployed without the need for specialized computing resources.

SECURE MULTIPARTY COMPUTATION

A secure multi-party computation is an interactive cryptographic protocol that allows for two or more mistrusting parties to jointly compute a function using their privately contributed data as input [26, 27]. In a resource constrained environment where techniques like homomorphic encryption are considered to be computationally expensive to be deployed to compute a function, secure multiparty computation is adopted to simulate the privacy-preserving implementation of the function using an interactive protocol. Usually, with secure multiparty computation protocols, the output or result of the protocols are can be considered to be public, unlike the private input data. Secure multiparty computation protocols are also computationally cheaper than homomorphic encryption, but perform worse with respect to communication overhead.

GARBLED CIRCUITS

Garbled circuits is an example of multiparty computation which supports secure two-party communication between *Alice* and *Bob*. At the root of this method lays a shared function $x = f(a, b)$ with input a from Alice and b from Bob. After the protocol execution has been completed, *Alice* and *Bob* both know x but neither of them learned other private information about the input of the other [28, 29].

PRIVATE SET INTERSECTION

Given two sets X and Y, set intersection expressed as $X \cap Y$ returns the common terms in both sets. A private set intersection (PSI) protocol achieve a similar goal only with the sets considered to be private. $PSI()$ is a private set intersection if for all $i \in PSI(X, Y)$, $i \in$

$X, i \in Y$. PSI protocols can be executed in a peer-to-peer model or a server-client architecture to determine the intersection of their inputs [30]. Private Set Intersection Cardinality (PSI-CA) is a variant of the regular PSI protocol between a Server and a Client. The Client only learns the size of the intersection between the input sets [30]. Authorised Private Set Intersection (APSI) is a different improvement of the regular PSI protocol between a Server and a Client. The Client learns the intersection between the Server and Client inputs with an additional constraint (Authorisation) that certain elements of Client's choice can be ignored during the intersection [31].

ORDER PRESERVING ENCRYPTION

Order Preserving Encryption (OPE) is a deterministic encryption scheme whose encryption function preserves numerical ordering of the plaintexts [32]. Consider two sets $X, Y \subseteq \mathbb{N}$ where $|X| \leq |Y|$, a function $f : X \rightarrow Y$ is said to be *order-preserving* if for all $i, j \in X$, $f(i) > f(j)$ holds for all $i > j$. This makes OPE suitable for query operations on a database even after encryption. Equality and range queries including operations such as COUNT, MIN, MAX, SUM, GROUP BY and, ORDER BY can then be applied on data protected using order preserving encryption [33]. OPE has found it application in genome data processing where it enables the encryption of the positions of short partial DNA sequences and preserves the numerical ordering of the plaintext positions [34].

3.1.3. GENOME DATA PROCESSING

Before research can be performed on a person's genomic data, DNA needs to be sequenced from the individual after which a digitised version of the genome also known as *in silico* genome is returned to the client who ordered it. Depending on the type of tests and the infrastructure being used by the research lab, various actors will handle the data with numerous operations. A simplification of how this process works is depicted in Figure 1. Throughout this whole process it is necessary to preserve the privacy of the individual's genome while limiting the decrease in accuracy and capabilities of processing the data. In the remaining part of this work, we will clearly define the settings that we have considered and also the genome data processing phases that are of interest to us. We will present the trade-off between privacy of the data and utility of the resulting privacy-preserving computation.

SCOPE OF STUDY

To adequately analyse the privacy of the different techniques applicable to genomic data, we decided to limit ourselves to specific phases of the process. Concretely, our literature research is focused on privacy-preserving techniques that can be deployed to protect the *in silico* genome, with the assumption that the genome could be resident with the owner, or stored as part of a dataset including multiple individuals' data to be used for genome wide association study (GWAS). Our considerations are limited to processes peculiar to step 3 and step 4 of Figure 3.1. For ease of reference we introduce an entity named *Alice*. *Alice* will be the individual who owns the genomic data where a data owner is a part of the protocol. The privacy challenges in these phases considered are the following:

- From the step where the *in silico* genome is returned to the *Alice*, the storage, transmission, and computations involving the genome is required to be protected

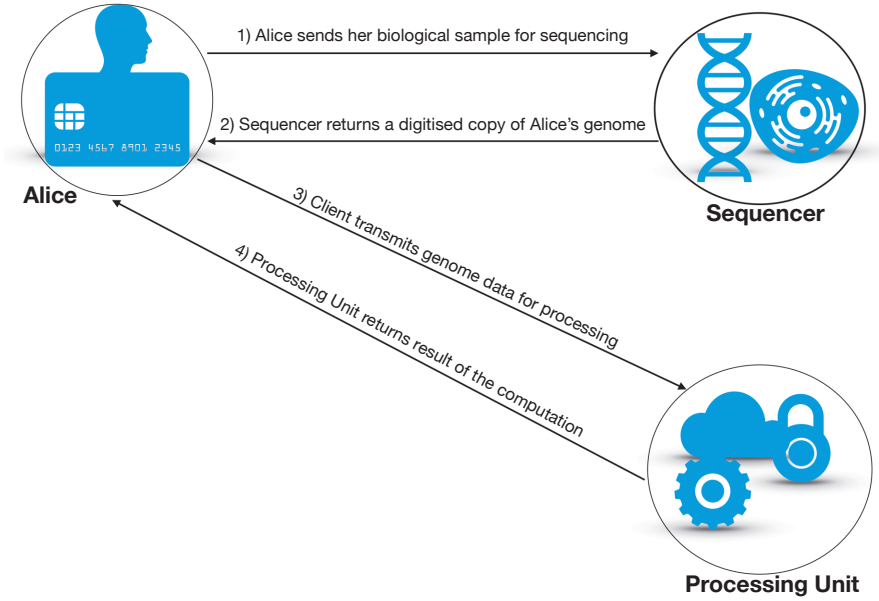

Figure 3.1: Overview of genome data processing life cycle

against privacy-threats.

- The *Processing Unit* who is responsible for running computations with the genome should not be able to deduce more information from the *Alice*'s genome than necessary. In an ideal situation, the *Processing Unit* gains no knowledge about *Alice*'s genome, however, in some situations it is necessary for this requirement to be relaxed, with the conditions clearly stated.

- In special scenarios, *Alice* should not able to obtain any knowledge about specific implementations of the processing techniques, such as algorithms or markers being used, in order to protect the intellectual property of the *Processing Unit*.

PRIVACY VS UTILITY

As mentioned previously, genomic data contains a lot of information about an individual, including privacy-sensitive information. One benefit of this characteristic is that genomic information can be used in a judicial and criminological setting to prove innocence or guilt of a suspect. To do this adequately, the tester needs to be certain that the genomic data being tested belongs to the suspect. This is regarded as integrity of the genomic data [11]. A drawback of this integrity requirement is that it adds a level of complexity to the process of preserving privacy when handling genomic data. Although information in the genome needs to remain private, the *Processing Unit* needs to be able to check whether or not it belongs to the person of interest, thereby leading to a privacy

versus utility question. On the other hand, where privacy measures are not taken to protect against the possible abuse of processing genome data, there are clear and present privacy risks[7]. Therefore, while privacy-preserving techniques are relevant to process genome dataset, only those solutions that do not jeopardize the utility of the data or results of the computations are relevant for considerations. We can conclude by saying that in considering the choice of privacy enhancing technique for genome data processing, utility and privacy are inversely proportional and the is always the need to find an acceptable trade-off.

3.1.4. GENOMIC PRIVACY RESEARCH

In order to bound the search space for our literature study, we are able to provide select protocols that involve one or more privacy preserving techniques for performing some form of genome data processing on genomic data. All protocols share a common assumption of targeting protection for the electronic version of the genome as described in Section 3.1.3, but they still contain settings-specific assumptions and constraints about the scenario in which their technique apply. These differences in scenarios should be taken into account when analysing and comparing the various techniques. As such, we will first give a brief description of the protocol setting and then we will explain the results of the techniques for the protocol.

PRIVACY-PRESERVING PROTOCOLS

Every scenario adopts different techniques and various techniques have their own pros and cons. We mainly address the practicality of each protocol in terms of computational complexity, communication complexity, storage cost, run-time and privacy guarantees of the techniques used.

Protocol X — Lauter et al. in [35] propose a privacy-preserving genome data processing protocol. In their setting, the aim is to compute a set of defined statistical algorithms over genome dataset outsourced to a cloud infrastructure, and this is designed within the *honest-but-curious* security model. Genotype data from multiple individuals are collected and encrypted using a homomorphic encryption, then stored in a cloud server where GWAS compuations are to be carried out using this dataset. Algorithms like *minor allele frequency (MAF)*, Cochran-Armitage Test for Trend (CATT) and Pearson Goodness-of-Fit test, and Linkage Disequilibrium are computed using the encrypted dataset. Lauter et al. introduce an entity who is responsible for managing keys, including key generation and ciphertext decryption where necessary, with the assumption that this party does not collaborate with other parties in pursuit of learning private information on the genome data. After the genotype data have been encoded and encrypted, a researcher can then be given access to utilize the dataset for computing their choice of statistical algorithm for which the protocol accommodates.

Result: This protocol demonstrate the feasibility of utilizing levelled homomorphic encryption for the preservation of privacy of outsourced data. The privacy requirement is guaranteed with the use of homomorphic encryption, and the encoding technique of data packing is introduced to enhance performance. The choice of parameters in the prototype achieves 80-bits of security and the implementation environment of include

Intel(R) Core(TM) i7-3770S CPU @ 3.10GHz, 8GB RAM, running 64-bit Windows 8.1., with the worst runtime for an algorithm recorded as 6.85 seconds.

Protocol A — Assume we have a testing party *Bob* that offers operations for Minor Allele Frequencies (MAF), χ^2 statistics and Hamming distance. Then there is an individual, *Alice*, who can make use of *Bob*'s resources to perform such operations on her own genomic data. In order to do so, *Alice* sends her digital version of her genome to *Bob*. To ensure privacy of her data, she first encrypts her genome via a homomorphic encryption scheme. For *Bob*, however, to be able to perform his operations as efficient as possible, he requires *Alice* to make use of a different homomorphic encryption for each of the three operations such that he can design his operations in a specific way for it to adequately perform calculations on the encrypted version of *Alice* her genome. Hence, *Alice* needs to adopt her encryption dependent on the test she desires to be performed on her genomic data. This method was developed by Bouti et al. [36]

Result: The technique used in scenario A was homomorphic encryption. Because of this, full privacy is ensured for *Alice*. Calculations performed on homomorphic encrypted data are most often slow due to its construction. However, since *Alice* needs to use a different homomorphic encryption based on which tests she wants to be performed, the calculations can happen more efficient for each test. Albeit this performance enhancements, there still exists some overhead in communication costs. The trade-off being made for more efficient calculations is that *Alice* has to adopt her encryption based on the tests which lowers the practicality of the method.

Protocol B — Sean Simmons and Bonnie Berger proposed a privacy-preserving protocol for computing GWAS algorithms such as Pearson Goodness-of-Fit test [37]. The setting in this protocol is as follows: *Bob* is a researcher who wishes to conduct a study using genomic data stored in a large genome database such as a *Cloud Server*. Bob makes a query to the database, asking for a Genome Wide Association Studies computation relating to a genotype-trait. An example of such computations could be the neighbour distance mechanism for identifying significant SNPs based on a chosen threshold. To ensure privacy in such a situation, Simmons et al. deploy differential privacy for the perturbation of input dataset or the result of the query before sending the differentially private result to *Bob*. This approach sure significant accuracy for the neighbour distance mechanism problem [37].

Result: This method showed an improvement not only in performance, but also in accuracy, compared to existing techniques. However, while this differential privacy protocol does not introduce computational complexity, it does suffer from noised results, due to the perturbation. Also, the privacy guarantee is not provably secure. Because of the error present in the results of the GWAS computations, this solution might not be suitable for very sensitive scenarios where lives are involved, thereby contributing to its drawback.

Protocol C — A secure genome testing protocol proposed by Cristofaro et al. for pri-

vate substring matching represents another privacy-preserving genome data processing protocol [38]. In this setting, *Alice* is holds a digitized genome which she wants to keep private, while *Bob* has a set of DNA markers. *Alice* uses her private genome data as input to a secure two-party computation protocol for conducting a testing, while *Bob* contributes his markers as input. These markers are used in medical tests to check if certain nucleotide blocks are present in a person's genome. For example, the markers for diseases such as diabetes are known, thereby making it possible for *Alice* to check her digitized genome matches the known markers. To do so, while maintaining her privacy, she sends a homomorphic encrypted version of her full genome to *Bob* who then can perform the tests with his DNA markers. When done, *Bob* sends the results back to *Alice* who can subsequently decrypt them and obtain the information she needed. Because *Bob* makes use of DNA markers, three additional privacy issues are raised from the perspective of *Bob*: 1) the position of the tested markers should remain secret to *Alice*, 2) the number of markers used in the tests should remain secret to Alice, and 3) in case the overall result turns out negative, the subset of the markers that match should also remain secret to *Alice*. To handle these issues, Cristofaro et al. proposed a protocol using the additively homomorphic variant of ElGamal cryptosystem [16].

Result: Although the situation described opposed some new constraints in terms of privacy for the tester, the proposal by Cristofaro et al. is able to handle them adequately. The results of the paper show that both *Alice* and *Bob* achieve their privacy requirements. That is,*Bob* does not gain more information out of *Alice*'s data, while *Alice* does not get more information about *Bob*'s DNA markers. Additionally, the implementation of this technique was shown to complete this two-party computation on a full human genome in less than 24 hours when implemented on the following environment: C++, Ubuntu12.10, with an Intel i7-3770 3.4GHz quad-core CPU and 16GB of RAM. We implemented 1024-bit AH-ElGamal. The downside, however, is that encryption of the data is rather slow (although this only needs to happen once) and sending the data through a network from *Alice* to *Bob* also adds some overhead to the protocol.

Protocol D — Wang et al. [39] propose a privacy-preserving pattern matching of genome sequence in the encrypted genome. In their proposed setting, *Alice* consults a sequencing lab to sequence her DNA with the goal of performing tests on her genomic data on a later date. When the sequencing is done, the lab encrypts the digitised version of *Alice*'s genome and outsources it to a *Cloud Server*. Now, only authorised parties can submit a test request to the *Cloud Server* to get permission for performing tests such as genome sequence pattern matching on an individual's genomic data. Such request can only be submitted if it is permitted by the owner of the data, that being *Alice*. For now, the encryption can only handle approximate sequencing matching tests. When the tests are performed, the authorised party receives the results from the *Cloud Server* which it can then forward to *Alice*. The encryption used on *Alice*'s data is a modified version of the Predicate Encryption (PE) scheme, developed by Wang et al. [39].

Result: In this protocol which is designed in a honest-but-curious security model, the storage of *Alice*'s genome gets outsourced to the *Cloud Server*. Here, her data is en-

crypted and can only be used for tests when permission is granted by *Alice*to an authorised party. Although only one type of test can be done by the authorised party, it does not give *Alice*a full privacy guarantee. The result is still shared with the tester and it is unclear whether more information can still be adopted from this technique. Computational wise there is no significant performance improvement from existing proposal, but by outsourcing the data only one round of communication is needed thereby decreasing the communication overhead.

Protocol E — Baldi et al. proposed a privacy-preserving protocol for computing a suit of tests [40]. In the protocol setting, *Alice* stores her personal genomic data at a data centre and wishes to have Paternity tests, Personalised medicine, or genetic compatibility tests performed on her data by agents ranging from personal physicians, to family members, pharmacies, hospitals, insurance companies, employers and government agencies (e.g., the FBI), or international organisations. This proposal opts for encrypted data, and because the encryptions are designed in such a way that different properties remain intact for different kinds of tests. Baldi et al. investigated which techniques works best for the different tests [40]. For paternity test, Private Set Intersection Cardinality (PSI-CA) was preferred, with inputs being the genomic data of *Alice* and that of the other person with whom test is conducted. PSI-CA is a technique which allows *Alice*to learn the properties *Alice* has in common with the other party, while this other party learns nothing about the result of the test. For personalised medicine tests, the Privacy Preserving Personalised Medicine Testing (PPPMT) technique is used. Genetic tests are performed with Privacy Preserving Genetic Compatibility Test (PPGCT) as described in [40].

Result: First for paternity testing, with PSI-CA the tester only learns the magnitude of ancestry rather than the contents. The downside of this technique is that it is very computational intensive because the number of nucleotides in the human genome is huge, at 3 billion base pairs, however, full privacy is ensured. Because PSI-CA has the performance drawback Baldi et al. [40] also considered using letters.Restriction Fragment Length Polymorphisms (RFLPs). RFLP is much faster because the DNA sample is broken into pieces by restriction enzymes and takes about a minute to compute on a computer with an Intel Core i5-560M (running at 2.66 GHz). The PPPMT protocol is reported to have a runtime of about 200 minutes on the same computer while PPGCT has a runtime of 67 minutes.

Protocol F —

The edit distance between sequences A and B is defined as the minimum number of edits (insertion, deletion, or substitution of a single character is counted as one edit) to change A into B. Wang et al. uses the concept of edit distance to propose a privacy-preserving protocol for querying patient similarities. Assume *Alice* has her medical file stored with her hospital that is also part of a larger network of hospitals. A copy of *Alice*'s genome is contained in this medical file. Now suppose this network of hospitals share a database containing all the genomes of each hospital provided with some additional information, i.e. when a genome belongs to a person with breast cancer. It is then possible for a tester of that hospital, say *Bob*, to compare genomes of a patient with the

genomes stored in the shared database. Such comparison can be performed in the way of calculating the edit distance of two genomes. Wang et al. developed a method of executing such calculations on full genomes while preserving the privacy of every patient in the database. [41] Their main idea is to use garbled circuits together with clustering different 'types' of genomes. Concretely will every hospital order their genomes in similar clusters such that tester *Bob* only needs to check the centre of each cluster to find related genomes to that of its patient. When the most similar matches are found, the tester can compare deeper into the cluster from which the matching genomes originate.

Result: In this protocol, a network of hospitals is described to share genomic data amongst each other. By using the garbled circuit protocol, Wang et al. were able to ensure privacy for *Alice* her genomic data in the hospital database. [41] By working with clusters, the authors were able to speed up to computation necessary for the tests. Additionally, the method proposed in the paper is able of handling a full human genome instead of just segments. A downside of the technique is that this solution utilized an approximation technique, thereby introducing errors in the results, although in a rather small percentage of the tests.

Protocol G — Restriction Fragment Length Polymorphisms (RFLPs) refers to a difference between samples of homologous DNA molecules that come from differing locations of restriction enzyme sites, and to a related laboratory technique by which these segments can be illustrated. In RFLP analysis, the DNA sample is broken into pieces (digested) by restriction enzymes and the resulting restriction fragments are separated according to their lengths by gel electrophoresis. Thus, in short, RFLP provides information about the length (and not the composition) of the DNA sub-sequences occurring between known sub-sequences that are recognized by particular enzymes. Cristofaro et al. propose a privacy-preserving protocol to genetic testing using RFLP [42]. In the setting, *Alice* owns a digital sample of her own genomic data and is responsible for the storage. *Alice* has an app on her smartphone which can perform tests such as paternity test, ancestry test and personalised medicine tests on the genomic data stored on her phone. For instance, when *Alice* visits a medical institute, she can test her genomic data on-demand with her phone and show the results to the specialists, which allows the specialists to give her more personalised treatment based on the results. The mobile app designed by Cristofaro et al. performs RFLP-based paternity tests, SNP-based paternity tests, ancestry tests, gynaecological tests and personalised medicine tests [42]. For the paternity test, PSI is used to preserve the privacy of *Alice*Authorised Private Set Intersection (APSI) is the technique used for personalised medicine test.

Result: The PSI testing technique is recorded to run within seconds, but the storage requirements are high for smartphone standards. Alternatively they propose that storage and computation can be done outsourced to an online service. Unfortunately ancestry- and gynaecological tests do not run within seconds or minutes. However, such an app which can test genomes has some serious privacy concerns. For instance, the app can be reverse engineered and the implementations of the tests can be retrieved. Furthermore, encrypted genomic information on the device can be quite easily retrieved from

the devices without the user knowing it [43]. And most worrisome of all, malicious apps might eavesdrop on the test results with no way of knowing.

Protocol H — In this protocol, Ayday et al. propose privacy-preserving for storing and processing genome data [34]. The proposal setting is presented as follows: there are four parties: 1) *Alice*, the owner of genomic data. 2) Biobank, stores the genomic data of *Alice* and other clients in databanks. 3) *Bob*, a researcher who tests genomic data. *Bob*makes queries to the Biobank to be able to perform the tests. 4) To ensure security they introduce a party named Mask and key manager (MK). This party grants access to the data of *Alice* in the Biobank. Which specific genetic tests to be performed are not considered in this scenario, but instead Ayday et al. considered security techniques for three possible kinds of attacks: A) a hacker or malicious Biobank employee who tries to perform genetic tests on genomic data in the Biobank. B) A hacker or a malicious MK employee who tries to infer genomic sequence from information provided by the Biobank, and perform genetic tests. C) A hacker or a curious colleague of *Bob* who wants to obtain private genomic data of a *Alice*. To guarantee privacy against possible data theft, order preserving encryption (OPE) technique is employed. This encryption masks the positions of the letters and preservers the numerical order of the plaintext positions, allowing to query segments of the genomic data.

Result: The encryption considered in this protocol is not very private because although the information is safe from theft, *Bob* does get to know the result of the test. Skepticism arises from the fact that only order preserving encryption (OPE) is used. This means that also the Biobank gets to know the result if the result is sent back to the Biobank. A drawback of the OPE encryption is that when a lot of queries are done, the person behind the genome can get identified. The Biobank stores data encrypted and can be queried, data transmission is fast and only little parts of data is sent. This approach requires the Biobank to store some overhead information because the positions of all segments and the list of nucleotides they accommodate has to be stored. Finally on their prototype implementation, querying segments of nucleotides takes approximately 5 seconds on an Intel Core i7-2620M CPU with a 2.70 GHz processor under Windows 7.

Protocol I — Again Hasan et al. proposed a protocol for secure count query over encrypted genome data [44]. In the protocol setting, four parties: 1) *Alice*, the owner of genome data. 2) A Certified Institute *(CI)* that is responsible for the storage of the genome data of clients such as *Alice*. 3) *Bob* is a researcher who performs relevant tests such as the susceptibility test to a disease. 4) A *Cloud Server* that offers services such as storage of encrypted genome data and processing data using algorithms that are available to it. The *CI* sends the subject's genome data to the *Cloud Server*, while *Bob*submits the list of relevant tests to the *CS*. The **CS** processes the test of *Bob* on the genomic data from *Alice*. The result is sent back to *Alice* who is the test subject. Hasan et al. addresses the challenge of maintaining privacy for *Alice* and for the researcher. To do this *Alice*is required to store her genomic data homomorphically encrypted with the *Cloud Server*. The researcher encrypts the test by converting it to garbled circuits.

Result: The CS can perform the garbled circuit test with the homomorphic encrypted genomic SNPs. The output is only readable by *Alice*, who has the decryption key. *Bob* knows that the algorithm is safe because the CS can not read this as well due to the security guarantee provided by the garbled circuits technique. The *Cloud Server* only performs the test and learns nothing from either *Alice*'s genomic data, test result. The data is modelled in an index-tree for every SNP, tree building can take from 10 seconds for 60 SNPs up to 1 minute for 300 SNPs on an Intel Core i5 3.3GHz processor with 8GB RAM on Ubuntu Linux 16.04. This protocol does not take into account what kind of tests can be performed on the SNPs. This means that we can not say how long *Alice*has to wait for a result when she requests a test on her genomic data.

Protocol J — Also, Namazi et al. proposed a privacy-preserving protocol for disease susceptibility testing [45]. In the protocol setting, they assume that *Alice* wants to perform susceptibility tests using her private genomic data. To do so, she has her genome sequenced and encrypted at a trusted certified institution. This institution uses lattice-based Somewhat homomorphic encryption (SHE) to generate and distribute keys among three parties: *Alice*, *Medical Center*, and *Cloud Server*. Additionally, *Alice*receives a Bloom Filter which denotes the SNPs present in her genome. The institute sends those SNPs involved in the test encrypted to the cloud service provider. At this point all parties are in possession of the required information to run the required susceptibility test. The test then commences with *Alice* receiving the genome loci with which to compute the test from *Bob* at the *Medical Center* and the cloud service provider receiving their corresponding contributions to the disease. Next *Alice* tests which locations are present in her genome using the Bloom Filter and sends them encrypted to the cloud service provider if present or a dummy location otherwise. At this point the cloud service provider can run the susceptibility test and sends the encrypted outcome to the *Medical Center*. At last the *Medical Center* decrypts and interprets the results.

Result: The technique used in this protocol improved on its predecessors by removing the need for sending back and forth encrypted information to be re-encrypted with the key of another entity. This is done by using a key-switching algorithm. The overall efficiency is hereby improved because less communication is required. The computational load is concentrated on the cloud service provider, since *Alice* and *Bob* solely need to encrypt input and decrypt output. In terms of computational performance it does not significantly improve from previous protocols. With respect to privacy, this technique allows for an access policy managed by *Alice* to be deployed on the encrypted data, which increases her privacy.

Protocol K — Danezis and Cristofaro proposed a privacy-preserving protocol for disease susceptibility testing. Assume that *Alice* wants to perform susceptibility tests on her genomic data. A trusted certified institution receives her genetic sample, sequences it and produces an encrypted encoding of all possible SNPs. The institution sends the entire encrypted sequence to a cloud service provider for storage. *Alice* receives a smartcard containing a partial secret key for decrypting her genomic data. The test can now be initiated by *Medical Center* at the medical centre following one of two variants pre-

sented by Danezis and Cristofaro [46] on a protocol previously proposed by Ayday et al. [47]. In variant 1, the *Medical Center* encrypts weights of the SNPs to test and sends them to *Alice*'s smartcard. All weights are checked with their corresponding SNPs, provided by the *Cloud Server*, in a streaming fashion. Afterwards a signature is checked to ensure a valid test was performed. The result is sent to *Medical Center* at the medical centre for decryption. In variant 2, based on secret sharing, the encrypted SNPs are loaded by both *Medical Center* and the *Cloud Server*. Both use them to compute sums over elliptic curve points. Both ciphertexts are sent to *Alice*'s smartcard to be added and decrypted before looking-up the result.

Result: Two major enhancements are presented in this protocol. These enhancements remove a limitation imposed by the protocol proposed by Ayday et al. [47]. This limitation being the leakage of which and how many SNPs were tested. Thus, the privacy is increased by this protocol. With respect to computational performance, one enhancement increase it by proposing an alternative encoding for SNPs and by making the computation to be executed on *Alice*'s smartcard more efficiently. The second enhancement uses secret sharing to reduce the online computation even further. Other improvements offered by this proposal over the existing solution is that of efficiency. The introduction of an additively homomorphic elliptic curve based El-Gamal scheme eliminates the huge computational costs associated with a proxy re-encryption implementation designed by Ayday et al. [47]

Table 3.1: Existing privacy-preserving protocols with their properties.

Protocol	Parties	Tests performed at	Tests	Techniques
A	Alice, Bob	Bob	MAF, χ^2, hamming distance	homomorphic encryption
B	Alice, Bob	Bob	neighbour distance	differential privacy
C	Alice, Bob	Bob	DNA markers	homomorphic encryption
D	Alice, Bob , Cloud service	Bob	approximate sequencing matching tests	predicate encryption
E	Alice, Bob	Bob	Paternity-, Personalised medicine- and genetic tests	PSI-CA, PPPMT, PPGCT
F	Alice, Bob , Hospitals	Bob	edit distance	garbled circuits
G	Alice	Alice	Paternity-, Ancestry- and personalised medicine tests	PSI, APSI
H	Alice, Bob , Biobank, MK	Bob	personalised medicine tests	OPE
I	Alice, Bob , CI, CS	CS		homomorphic encryption, garbled circuits
J	Alice, Bob , medical centre, CS	CS	SNP tests	homomorphic encryption
K	Alice, Bob	Bob	susceptibility test	private key encryption, elliptic-curve cryptography
L	Alice, Bob, CI, CS	CS	paternity test	commutative encryption
M	Alice, Bob , CS	CS		homomorphic encryption
N	Alice, CS	CS		Obfuscated bloom filter,
X	Alice, CS			homomorphic encryption

Protocol L — Lei et al. attempts to propose a privacy-preserving protocol where 2 parties can jointly compute a paternity test, using their private genome data [48]. The setting is as follows: suppose *Alice* wants to perform a paternity test with another individual named *Bob*. They register at a certificate authority which generates keys and pseudonyms for both to ensure their privacy. *Alice* and *Bob* submit their genome sample to a trusted certified institution to be digitised and encrypted. The institution scrambles the order of the data and stores this at a *Cloud Server*. When the *Cloud Server* has received requests from both *Alice* and *Bob* for paternity testing it verify their integrity before proceeding with the protocol. The cloud service provider asks the certificate authority using the pseudonyms for an scrambled matrix containing the indices of the data to compare. With this matrix the cloud service provider can now execute the test on the encrypted data and share the result with *Alice* and *Bob*. Lei et al. claim to use the property of Commutative Encryption to achieve this protocol description [48].

Result: While the problem definition appears to be clear, the underlying security construction of their proposed solution appears to be flawed. For example, the idea of commutative encryption as describe in this work assumes that RSA encryption scheme is to be adopted. And parameters for *Alice*and *Bob*will have the following properties:
(p_A, q_A) : *Alice*'s RSA secret primes,
(p_B, q_AB) : *Bob*'s RSA secret primes,
with $p_a \neq p_B$ and $q_A \neq q_B$, but somehow they claim that $p_A \cdot q_A = p_B \cdot q_B$. thereby achieving commutative encryption property.

Protocol M — A decentralized protocol for secretly searching for nucleotide on multiple databases was proposed by Yamamoto and Oguchi [49]. The protocol setting assumes that *Alice* wants to perform a string search on multiple digitised genome databases. A cloud service provider assigns a machine to act as a "master" and a couple others as "workers". This master communicates with *Alice* and coordinates the distribution of workload among the workers. The workers perform the actual computations involved using fully homomorphic encryption scheme. *Alice* encrypts her query and sends them to the master, which handles it and sends the result back to *Alice* afterwards.

Result: With respect to the protocol description, the privacy of the users whose digitised genomes are being queried by *Alice* appears to be protected. However the practicality of the proposal is not entirely clear. For instance, the computation that happens at the worker node is not clearly defined. How keys are managed and comparison of query nucleotide against the database nucleotide strings are not described in details. This poses the question “What function is being computed and what are the underlying operations needed to realize it?”

Protocol N — Using Bloom filters and homomorphic encryption, Perl et al. proposed a privacy-preserving protocol for confidential queries over genome databases [50]. Suppose *Alice* wants to execute a query on genomic data, but she lacks the resources to do this herself. A cloud service provider can provide services for the required resources, but *Alice* does not want the query or the result to be known by the cloud service provider.

She submits her query encrypted to the cloud service provider. First, the cloud service provider reduces the search space by using Obfuscated Bloom Filters provided by *Alice*. Then, it uses a homomorphic encryption scheme to perform the query on the reduced search space. The result is returned to *Alice* who can decrypt it.

Result: Concerning the practicality, this method is as fast as comparable algorithms with no concern for privacy. This enables it to be used in real world applications; an actual implementation is provided to the public for download. Additionally, the method can handle both discrete data sets and streamed data. The method enables the trade-off between performance and privacy ratio to be adjusted by selecting an appropriate security parameter since that determines the level of obfuscation and thereby the size of the intermediate result set.

3.1.5. CONCLUSION

DNA sequencing is a beneficial technique for scientific research and can provide many insights into the configuration of an organism. When applied to human DNA, a full genome can be generated containing the precise composition of an individual's nucleotide base pairs. While being very informative to scientists, this type of data should be handled with the highest level of privacy to avoid privacy incidences that might threaten the privacy of genome data owners. To protect the privacy of genome data owners while processing the digitised version of the genome, many techniques and approaches have been considered and applied with the goal of preserving privacy.

We have studied the recent literature on this topic with the goal of obtaining an answer to the question: what are the techniques utilized for preservation of privacy when managing genomic data? We divided this question into sub-questions, seeking to find an answer on which techniques are being used, on which genome data computation, and what parties are involved. This research concludes that multiple privacy-preserving techniques are being used for handling genomic data and that there is no 'one size fits all' technique currently available, as can be observed from Table 3.1. Most often, the adopted technique is dependent on the computation that needs to be performed on the data. We see that for statistical tests, differential privacy provides a good option for accuracy and efficiency, with a trade-off on privacy owing to the lack of provable guarantees. On the other hand, provable techniques such as homomorphic encryption provides better privacy guarantees but with the drawback of computational inefficiency as well as storage inefficiency attributed to data expansion. We also observe that not every technique currently proposed in the literature proves to be feasible. This is due to limitations in accuracy or performance overhead. Another limitation is the set of tests that can be performed with each technique. This is due to the fact that most techniques need to be adapted to the nature of the test. As a final conclusion, we see that performing tests requires resources like computational power, storage space, and efficient algorithms. These are resources not available to every individual which is why external parties similar to cloud infrastructures are essential in the processing of genomic data.

3.2. PRIVACY SAFE LINKAGE ANALYSIS

Genetic data are important dataset utilised in genetic epidemiology to investigate biologically coded information within the human genome. Enormous research has been delved into in recent years in order to fully sequence and understand the genome. Personalised medicine, patient response to treatments and relationships between specific genes and certain characteristics such as phenotypes and diseases, are positive impacts of studying the genome, just to mention a few. The sensitivity, longevity and non-modifiable nature of genetic data make it even more interesting, consequently, the security and privacy for the storage and processing of genomic data beg for attention. A common activity carried out by geneticists is the association analysis between allele-allele, or even a genetic locus and a disease. We demonstrate the use of cryptographic techniques such as homomorphic encryption schemes and multiparty computations and show how such analysis can be carried out in a privacy friendly manner. We compute a 3×3 *contingency table, and then, genome analyses algorithms such as linkage disequilibrium (LD) measures, all on the encrypted domain. Our approach to this computation guarantees privacy of the genome data under the semi-honest security settings, and provides up to* 98.4% *improvement on computation time and storage, compared to the state-of-the-art solution.*

Parts of this chapter have been published as:
(1) Ugwuoke, C., Erkin, Z., & Lagendijk, R. (2017). Privacy-safe linkage analysis with homomorphic encryption. *In IEEE 2017 25th European Signal Processing Conference (EUSIPCO).*

3.2.1. INTRODUCTION

In recent years, it has become possible to perform whole human genome sequencing, which was not an easy feat only a few decades ago [51, 52]. Geneticist and researchers are now depending on the availability of the human genome in digital form to conduct ground breaking research. The genome contains rich information about the direct owner, relatives and even species, and most of these information are yet to be properly understood by scientist [51, 53]. Therefore, it is often the case that the genome is investigated to obtain various types of relationships. These relationships may include paternity relationships, possible human emigrations hundreds of years ago, gene-phenotype relationships, gene-disease relationships, patients response to a particular medication. Investigating relationships as listed above have numerous advantages which would include better understanding of human genome and possibly provide for better preventive and personalised healthcare [51, 52, 54]. However, the genome is a very sensitive data, which contains lots of other information about the owners, thereby posing a privacy threat to those who provide their genomes for various scientific or medical activities [52, 54].

When analysing gene-disease relationship, two conditions are feasible. First, a genetic marker could have a direct effect on a disease, thereby said to have a causal relationship type of association with the disease, and the marker doubles as the disease locus. Alternatively, a disease locus could be in linkage disequilibrium with a genetic marker, hence an indirect gene-disease association. In the latter type of association (which is our mode of interest in this paper), the genetic marker is not the same as the disease locus, and scientist often perform computation of statistical measures to ascertain the degree of LD, being, the threshold to confirm the suspicion of an association. It is not often the case that a single gene is responsible for a specific trait (disease, phenotype) and a known way for investigating gene-disease association could be to compute LD measures between a known genetic marker and a suspected allele. However, when individuals donate their genome as sample for analyses, it is usually common to re-identify participants, creating a huge privacy-risk [54, 55]. The challenge then become, whether we can perform computational analyses on genome data, without compromising the privacy of the genome owners.

We consider a scenario where a processing entity with sufficient resources store and process genome data, and an authenticated researcher seeks to compute an operation over the data stored by the processing entity. The data resident with the processing entity may have been voluntarily contributed by individuals, or provided by some verified medical institution. The researcher is in need of computing an LD statistic measure over the rich dataset resident with the processing entity. Because of the privacy-sensitive nature of the genome, it has become necessary to adapt privacy-preserving measures while performing computations that require genome data. Proposed solutions have suggested obfuscating the genetic data using different approaches like statistical data anonymization techniques and secret sharing [55]. Other studies have also suggested access control and security of databanks as the only measure, but that does not protect the privacy of the participants. The studies [35, 56] recommend cryptographic solutions like homomorphic encryption (HE), to encrypt the data and homomorphically compute the LD statistic measures. Deploying encryption and related cryptographic techniques are

preferred because it is easy to mathematically prove the security of cryptosystems and equally extend proofs to the constructed solutions, therefore one is able to estimate possible information leakage. In our work we propose a cryptographic solution, by encrypting the data and computing the relevant statistical analysis over the encrypted data. We treat the secure databank of genomic data as a secure signal sequence, which needs to be outsourced to an untrusted processing entity for computation analysis. The encrypted signals (genetic data) are sensitive and need to be processed in a manner that pays keen attention to privacy of the data. With the encrypted data being transmitted and processed by an entity who only has the computational resource but not able to learn the content of the data, it equally demonstrates that encrypted signals can be generated and processed without threat to privacy.

Our Contribution: We are proposing an efficient method for computing LD statistic measures over encrypted genome data. We adopt HE as a technique to securely store and privately compute LD measures from a genotype databank, owing to the provable security and privacy guarantees it provides. Furthermore, we introduce an honest-but-curious Key Manager for our solution, in order to improve efficiency of computing parameters and reduce storage costs by 83.3%. Also, we adapt the genotypic LD approach of computing the LD measures as against the allelic LD approach, because the allelic LD approach requires haplotype estimation techniques, which is computationally expensive and often bias [57, 58]. We adopt data packing technique to help manage the data expansion challenge that comes with encrypting the genes. Our encoded data storage and retrieval design makes it easier to dynamically compute the contingency table parameters necessary for computing the statistical measures, which shows a significant improvement from existing works [35, 56] that adopted homomorphic techniques.

Outline: In the rest of this paper, Section 5.1.2 discusses some related literature, while Section 5.1.3 contains preliminaries relevant to our work. In Section B.3 we propose our solution for privacy-friendly computation of LD measures. Section 3.2.5 contains security and performance analyses and finally, in Section B.7 we have conclusion.

3.2.2. RELATED WORKS

Prior to our work, different authors have proposed various techniques for addressing privacy concerns in genome data processing, ranging from differential privacy, secret sharing and homomorphic encryption [59]. Specifically, Wu and Haven [60] demonstrated a secure computing of statistical analysis algorithms over encrypted data. Their work demonstrates the use of leveled homomorphic encryption to compute the mean and covariance of a dataset, other than genome data.

More recently, Lauter et al. [35] conducted a study which demonstrates the application of homomorphic encryption in the analysis of genomic data. The study shows that statistical algorithms (Pearson Goodness-of-Fit Test, r^2-measures of LD, Estimation Maximization (EM) algorithm for haplotyping, etc) peculiar to genetic studies can be replicated over encrypted data. Their solution aims to protect privacy of participants whose genome data are used in the analysis. The study however provides a solution that incurs three times the storage cost, due to their choice of design that maps a single gene value to three homomorphic ciphertexts. Lauter et al. also implement the construction of the 3×3 contingency table in a computationally expensive method.

Lu et al. [56] propose a solution which allows for genomic data to be securely outsourced to a third party who should perform the analysis over the encrypted dataset. They utilised leveled homomorphic encryption [22] and present a result that outperformed Lauter's implementation [35]. They deployed data packing techniques thereby reducing the three ciphertext to one gene mapping that was suggested in [35]. Lu et al. describe their work for chi-square test using allele frequencies, which is obtained from genotype/phenotype values contributed to the processing party.

Other works include that of Shahbazi et al. [61], whose work presents secure computation of LD measures and Cochran Armitage Test for Trennd (CATT) using secret sharing. The adoption of secret sharing requires the use of multiple servers which is bounded by non-collusion assumption. A secret sharing solution allows for a faster computation but incurs more communication rounds and storage requirements.

3.2.3. PRELIMINARIES

There are 4 major entities in our description, which are 1) Storage and Processing Entity (*SPE*), 2) Researcher (*R*), 3) Key Manager (*KM*) and 4) Encoder. We loosely refer to whoever is responsible for the encryption of the genotype as encoder, this could be a participant or a verified medical institution. The *SPE* is responsible for storing all encrypted genome and subsequently performs computation on the genome on behalf of an authenticated *R* who is interested in computing an LD statistic measure over the dataset. The *KM* is an honest-but-curious entity who is only responsible for key generation, distribution and secure decryption of final computation results. Also, individuals whose genomes are available with the SPE are also called participants. Therefore, a participant's record is interchangeable with a sample.

LINKAGE DISEQUILIBRIUM MEASURES

We shall consider two hypothetical genetic markers X and Y, with each marker having two alleles of the same gene. Marker X has the alleles A and a while marker Y has the alleles B and b. Linkage Disequilibrium is said to exist when two or more alleles at different loci are observed to often be inherited together in a non random manner [57]. Statistical LD measures such as Pearson's correlation, Lewontin's D, linear regression are computable given counts of genotypes.

There are two ways of measuring LD statistics, *allelic* or *genotype-based*. *Allelic* LD measures require haplotype estimation techniques which is not trivial, but *genotype-based* approach allows computation without haplotype estimation. In our work, we have only the genotype data, hence the adoption of *genotype-based* approach for computing LD statistics measures. We model computations for: 1) The digenic LD between two markers X and Y, represented as $\mathscr{D}_{XY}$. 2) The Pearson's correlation coefficient, represented as Δ_{XY} . Table 3.2 is a 3×3 contingency table, which shows an example of genotype counts gen_{ij}, and their marginal sums. n_i , m_j represent the sum of all values in row i , and column j respectively, $i, j \in \{0,1,2\}$. For instance, $n_0 = a+b+c$, $gen_{00} = a$, also, $Q = n_0 + n_1 + n_2 = m_0 + m_1 + m_2$, which is the total number of participants. The contingency table shall form the basis of the rest of our computation.

Given that $P(\cdot)$ represents frequency and p_A is the frequency of the allele A, $\mathscr{D}_{XY}$ is computed as [57]:

$$\mathscr{D}_{XY} = 2P\left(\begin{array}{c|c} A & A \\ B & B \end{array}\right) + P\left(\begin{array}{c|c} A & A \\ B & b \end{array}\right) + P\left(\begin{array}{c|c} A & a \\ B & B \end{array}\right) + \frac{1}{2}\left(P\left(\begin{array}{c|c} A & a \\ B & b \end{array}\right) + P\left(\begin{array}{c|c} A & a \\ b & B \end{array}\right)\right) - 2p_A p_B\,, \tag{3.1}$$

and Eq. (3.1) can be estimated by,

$$\hat{\mathscr{D}}_{XY} = \frac{1}{Q}(a+b+d+\frac{1}{2}e) - 2\hat{p}_A\hat{p}_B\,, \tag{3.2}$$

$$\hat{p}_A = \frac{2n_0+n_1}{2Q}\,,\ \hat{p}_B = \frac{2m_0+m_1}{2Q}\,. \tag{3.3}$$

Let $g\hat{e}n$ be estimated genotype count. Then,

$$s_{xy} = \left(\sum_{i=0}^{2}\sum_{j=0}^{2} ij\cdot\frac{g\hat{e}n_{ij}}{Q}\right) - \bar{x}\bar{y}\,, \tag{3.4}$$

where,

$$\bar{x} = \sum_{i=0}^{2} i\cdot\frac{n_i}{Q}\,, \text{and } \bar{y} = \sum_{j=0}^{2} j\cdot\frac{m_j}{Q}\,, \tag{3.5}$$

and,

$$s_x^2 = \left(\sum_{i=0}^{2} i^2\cdot\frac{n_i}{Q}\right) - \bar{x}^2\,,\ \text{and}\ s_y^2 = \left(\sum_{j=0}^{2} j^2\cdot\frac{m_j}{Q}\right) - \bar{y}^2\,. \tag{3.6}$$

Given the above equations, the Pearson's correlation coefficient can be estimated as

$$\hat{\Delta}_{XY} = \frac{s_{xy}}{s_x\cdot s_y}\,. \tag{3.7}$$

Table 3.2: Genotype counts at two bi-allelic markers X and Y

	BB	Bb	bb	Σ
AA	a	b	c	n_0
Aa	d	e	f	n_1
aa	g	h	i	n_2
Σ	m_0	m_1	m_2	Q

HOMOMORPHIC ENCRYPTION

Homomorhpic Encryption (HE) allows for a simple operation to be performed on ciphertexts, such that the resulting ciphertext would decrypt to the same value as would be obtained if the algebraic operation were to be performed on the plaintext values. Let $E_{pk}(\cdot)$ and $D_{sk}(\cdot)$ represent encryption and decryption functions respectively. m_1 and m_2 are two messages and k is a scalar value, while $\oplus$ and $\otimes$ are arbitrary operations on the ciphertexts. Then, homomorphism is defined as follows,

$$D_{sk}(E_{pk}(m_1) \oplus E_{pk}(m_2)) \;=\; m_1 + m_2\,,$$

$$D_{sk}(E_{pk}(m_1) \otimes k) \;=\; m_1 \times k\,.$$

We leverage on the additive and scalar multiplicative properties of the HE scheme described by Paillier [15] to compute the required statistical algorithm over an encrypted genome dataset. We refer readers to [15] for more detail about the cryptosystem.

Secure Multiplication Protocol (SMP): The homomorphic cryptosystem [15] of choice only offers additive homomorphism and no multiplicative homomorphism, we therefore initiate a secure two-party protocol in order to obtain the multiplication of two encrypted values [62]. Given two parties $Alice$ and Bob, $Alice$ holds two ciphertexts $E_{pk}(m_1)$ and $E_{pk}(m_2)$ and requires to compute the product $E_{pk}(m_1 \cdot m_2)$. Bob holds the secret key sk, $Alice$ picks randoms $r_1,\ r_2$, and encrypts, then computes $E_{pk}(m_1 + r_1)$ and $E_{pk}(m_2 + r_2)$, then sends the masked values to Bob. Bob decrypts and multiplies the results, then encrypts $E_{pk}(m_1 \cdot m_2 + m_1 \cdot r_2 + m_2 \cdot r_1 + r_1 \cdot r_2)$ and sends to $Alice$. Finally, $Alice$ can unmask the value to obtain $E_{pk}(m_1 \cdot m_2)$.

3.2.4. PRIVACY-FRIENDLY LINKAGE ANALYSIS

Given the $Encoder$, SPE, R and KM setting, our aim is to preserve the privacy of the participants whose encrypted data are with SPE, and from which an LD-measure is to be computed. R is also guaranteed to obtain a correct computation result from the analysis, but should not learn any identifying information from the result. Our solution is twofold, first we homomorphically construct the 3×3 contingency table, as presented in Table 3.2, with each parameter computed as an independent ciphertext. Secondly, we use the parameters from the constructed table, as input to computing any LD statistic measure of choice. In the entirety of our protocol, there is a non-collusion assumption between any two entities.

The storage overhead of applying encryption is due to data expansion, therefore, we choose to efficiently constrain data expansion in our solution. For this reason, we introduce data packing technique for the encrypted values.

GENOTYPE ENCODINGS

To cushion the effect of data growth inherent in encrypting the genotype, the $Encoder$ performs the following one time operation to encode genotypes. Let N be plaintext size of the cryptosystem, recall that Q is the record size of the genotype counts from Table 3.2.

Setup: The KM generates cryptographic keys $(pk,\ sk)$ and makes pk public, and keeps sk secret. Let κ be the security parameter, $\ell = \lfloor \frac{\log_2 N}{\log_2 Q + \kappa + 1} \rfloor$, where ℓ is the number of $slot$ that are contained in the plaintext size of N. $slot_j$ represents the j^{th} slot in N and $j \in \{0, \ldots, \ell - 1\}$. The $Encoder$ reserves the last 4 slots $\{slot_{\ell-4}, slot_{\ell-3}, slot_{\ell-2}, slot_{\ell-1}\}$ for indicating genotype intersections, this means there are only $\ell - 4$ slots reserved for genotypes. Let number of genes to be encoded in a single ciphertext be $\delta = \frac{\ell-4}{3}$, which means that a single gene needs 3 consecutive slots, each of size $(\log_2 Q + \kappa + 1)$-bits. Let $\{X, Y, \cdots, Z\}$ be a set of markers.

Let t be the plaintext encoding, so that,

$$t = |X_0|X_1|X_2|Y_0|Y_1|Y_2|...|Z_0|Z_1|Z_2|a|b|d|e| \,. \tag{3.8}$$

For a given genotype database with maximum record threshold of Q, and each record $E_{pk}(t)$ being a ciphertext of δ-genes, SPE reconstructs a similar table as the model in Table 3.2. For a genetic marker X, with dual alleles A/a and possible genotype values of $\{AA, Aa, aa\}$ with corresponding index $\{0,1,2\}$ respectively. The $Encoder$ allocates a triple-slot to the marker X, with the genotypes mapped to the corresponding slot index, i.e X_i for $i \in \{0,1,2\}$ as indicated in Eq. 3.8. For every gene in a sample encoding, and given the participant's genotypes, the value 1 is entered in the slot for every corresponding genotype expressed by the participant, and every other genotype slot is completed with value 0.

Step 1: For two genetic markers X, Y in the encoding that are of interest for computation, the $Encoder$ indicates corresponding intersection slots with value 1 for where an intersection for one of $\{a = AA/BB, b = AA/Bb, d = Aa/BB, e = Aa/Bb\}$ exists, and value 0 otherwise. $Encoder$ then sends $E_{pk}(t)$ to SPE.

Step 2: To reconstruct Table 3.2 from a database of T records, with each record $E_{pk}(t_j)$ for $0 \leq j < T$ representing encrypted genotypes modelled after Eq. 3.8. SPE computes, $E_{pk}(genSum) = \prod_{j=0}^{T-1} E_{pk}(t_j) = E_{pk}(\sum_{j=0}^{T-1} t_j)$, as a single ciphertext courtesy of the Paillier cryptosystem [15]. It can be observed that the summation allows for slots to be summed component-wise, without overflowing into a neighbouring slot, therefore providing a ciphertext that should decrypt to a plaintext which preserves the encoding in Eq. 3.8.

Step 3: In reference to Table 3.2, SPE can obtain the following parameters $\{a, b, d, e, n_0, n_1, n_2, m_0, m_1, m_2\}$ as a packed ciphertext from the homomorphic addition result.

Secure Unpacking Protocol (SUP): An SUP requires that SPE initiates a secure two-party protocol with KM in order to unpack a single ciphertext of encoded sums of t's, into 10 independent ciphertexts of the parameters $\{a, b, d, e, n_0, n_1, n_2, m_0, m_1, m_2\}$, from which ciphertexts of the remaining variables can be obtained homomorphically. SPE chooses a cryptographic secure random number r of size N-bits, encrypts r and performs an additive masking of $genSum$ to obtain $E_{pk}(genSum + r)$, which is sent to KM for unpacking.

Step 4: KM decrypts the masked ciphertext to obtain a masked plaintext P, and splits P into ℓ parts, each of size $(\kappa + \log_2 Q + 1)$-bits, then encrypts each of $P_0, \cdots, P_{\ell-1}$ and returns the ordered values to SPE.

Step 5: SPE receives $\{E_{pk}(P_0), \cdots, E_{pk}(P_{\ell-1})\}$ and splits r into $\{r_0, \cdots, r_{\ell-1}\}$, each of size $(\kappa + \log_2 Q + 1)$-bits, encrypts each r_i for unmasking the corresponding P_i.

Step 6: To construct Table 3.2, *SPE* has the encryption of each of $\{a, b, d, e, n_0, n_1, n_2, m_0, m_1, m_2\}$, from which *SPE* deduces the rest of the variables as follows: $c = n_0 - a - b$; $f = n_1 - d - e$; $g = m_0 - d - a$; $h = m_1 - b - e$; $i = m_2 - c - f$; and $Q = n_0 + n_1 + n_2$. And the table structure in Table 3.2 is correctly constructed.

Step 7: When the numerators and denominators are computed homomorphically, the encrypted result is forwarded by the *SPE* to *R*, who is then required to further run a secure two-party computation with *KM* for secure decryption. Also note that for the algorithms that require homomorphic multiplication, *SPE* runs an *SMP* with *KM*.

HOMOMORPHIC COMPUTATIONS

Once the variables of Table 3.2 are all computed, any LD statistic measure which require only the available parameters as input, can be computed homomorphically by the *SPE* without learning the contents of the ciphertexts. For example, in order to estimate the Pearson's correlation coefficient $\hat{\Delta}_{XY}$ from our constructed table of ciphertexts, we rewrite Eq. 3.7 as follows;

$$\hat{\Delta}^2_{XY} = \frac{[Q\cdot(e+2f+2h+4i)-(m_1+2m_2)(n_1+2n_2)]^2}{[Q\cdot(n_1+4n_2)-(n_1+2n_2)^2][Q\cdot(m_1+4m_2)-(m_1+2m_2)^2]}$$
$$= \frac{[Q\cdot(e+2(f+h+2i))-(m_1+2m_2)(n_1+2n_2)]^2}{[Q\cdot(n_1+4n_2)-(n_1+2n_2)^2][Q\cdot(m_1+4m_2)-(m_1+2m_2)^2]}\,. \tag{3.9}$$

The above equation correctly computes the square of the Pearson's correlation coefficient using encrypted inputs. In the same way, we can rewrite Eq. 3.2 as:

$$\hat{\mathscr{D}}_{XY} = \frac{Q\cdot(2(a+b+d)+e)-(2n_0+n_1)(2m_0+m_1)}{2Q^2}\,. \tag{3.10}$$

According to [**?**], the goodness-of-fit statistic test for a locus can be re-written as:

$$\chi^2 = \frac{Q\cdot D^2}{p_A\cdot(1-p_A)} = \frac{Q\cdot(4Q\cdot n_0-(2n_0+n_1)^2)^2}{(2Q-(2n_0+n_1))(2n_0+n_1)^2}\,, \tag{3.11}$$

where,

$$D = P_{AA} - p_A^2\,. \tag{3.12}$$

3.2.5. SECURITY AND PERFORMANCE ANALYSES

SECURITY AND PRIVACY ANALYSES

Our aim is to provide privacy of genome data during storage and processing of the data, such that the utility of the data is not lost. For that, we recommend at least 80-bits of security, and for our chosen homomorphic scheme, we require at least 2048-bits of Paillier [15] plaintext size. Our choice of parameters can handle up to 100,000 samples. On the condition that no two entities within the protocol collude, we have:

Encoder: The *Encoder* performs a one time operation by encoding and encrypting the inputs. After which, he is not an active member of the protocol. Apart from the

Table 3.3: Computational complexity for contingency table.

	Enc.	Add.	Mult.	Dec
Lauter et al.	$6N$	$9N+14$	$9N$	0
Our proposal	$N+21$	$N+12$	0	1

values being learnt during encoding, the *Encoder* learns nothing else about other samples. However, a single individual should not be allowed to encode all data submitted to the *SPE*, because, such an individual will know the result of computations requested of the *SPE*. Finally, an *Encoder* should not know how much samples another *Encoder* submits to *SPE*.

SPE: Only ciphertexts are stored on the server, *SPE* does not learn the content of the data stored on his server, and does not also learn the content of the processed results, since it does not have the secret key. The *SPE* can however learn the total number of records, and can therefore deduce the value for Q, but this can be mitigated by allowing the *Encoder* add some dummy samples which are encryptions of zero. The dummy samples will not affect results of computations, but will only add a computation cost to generating Table 3.2, as well as storage cost to the server. Only the *SPE* is responsible for knowing the changes (insertion and deletion) of samples.

Key Manager: The KM is an honest-but-curious entity, and is only allowed to interact with masked values. Therefore, the KM does not learn the true values he is presented to operate on, so long as he does not collude with another entity.

Researcher: He obtains aggregated results such as numerator and denominator. He does not possess enough information to solve for the individual variables, therefore the privacy of the contributing samples are protected from the Researcher.

PERFORMANCE ANALYSES

To obtain a fair comparison of our proposal with Lauter et al's. [35], while preserving the same level of security offered by their work, we present here a C++ implementation of both approaches, using GMP library version 6.1.2 on a 64-bits Intel core 2 Quad @ 2.66GHz, running Ubuntu 14.04 LTS and Paillier cryptosystem, with 80-bits security for N samples. We also mention that one *SUP* requires 21 encryptions, 11 homomorphic additions and one decryption. Furthermore, we present a complexity comparison only for constructing the contingency table in Table 3.3, from which variables are used as inputs to compute LD measures and other genome analyses. We ignore the comparison for computing genome analyses algorithms because those are independent of the sample size.

It can be observed from Table 3.3 and 3.4 that our approach for computing the contingency table improves Lauter et al's. [35] proposal by 83% storage cost and 98.4% computational cost. Also we present the average runtime for 1000 runs of the cyrptosystems (Paillier, Microsoft Simple Encrypted Arithmetic Library) operations used in Table 3.4.

Finally, we present the complexity for various genome analysis algorithms we implemented in Table 3.5.

Table 3.4: Operations Timing results in seconds.

	Enc.	Add.	Mult.	Dec.
Paillier	0.02287	0.000012	0.14601	0.022738
SEAL	0.08990	0.000323	0.67082	0.087052

Table 3.5: Algorithm Complexity

	Add.	Sub.	Mult.	Scalar Mult.
Pearson's Corr. Coeff.	7	3	7	4
LD Coeffiecient	5	1	3	4
Goodness-of-fit Test	1	2	6	3

3.2.6. CONCLUSION

We present a secure, privacy-preserving and efficient approach for computing LD measures over genome data. Our approach provides significant improvements in storage and computational complexity from the existing work we compared with. We produce a 98.4% improvement for computing the 3×3 contingency table over that of Lauter et al. We introduce an efficient Key Manager in a semi honest security setting, who we leverage on to implement a secure and privacy-safe packing technique. We implement and show performance results for algorithms that use genome data, such as, Pearson's correlation coefficient, Goodness-of-fit test, LD coefficient. Our approach can also be used to compute other LD measures for which the equations are re-written to be executed using basic additions, subtractions and multiplications. Our construction is robust and can accommodate up to 100,000 samples for the parameters presented here.

3.3. PRIVACY-PRESERVING ASSOCIATION STUDY

The continuous decline in the cost of DNA sequencing has contributed both positive and negative impacts in the industry and research community. It has now become possible to harvest large amounts of genetic data, which researches believe their study will help improve preventive and personalised healthcare, better understanding of diseases and response to treatments. However, there are more information embedded in genes than are currently understood, just as a genomic data contains information of not just the owner, but relatives who might not subscribe to sharing them. Unrestricted access to genomic data can be privacy invasive, hence the urgent need to regulate access to them and develop protocols that would allow privacy-preserving techniques in both computations and analysis that involve these very sensitive data. In this work, we discuss how a careful combination of cryptographic primitives such as homomorphic encryption, can be used to privately implement common algorithms peculiar to genome-wide association studies (GWAS). This obviously comes at a cost, where we have to accommodate the trade-off between speed of computations and privacy.

Parts of this chapter have been published as:
(1) Ugwuoke, C., Erkin, Z., & Lagendijk, R. (2016). A Privacy-Preserving GWAS Computation with Homomorphic Encryption. *In 2016 37th WIC Symposium on Information Theory in the Benelux/6th WIC/IEEE SP Symposium on Information Theory and Signal Processing in the Benelux, Louvain, Belgium*

3.3.1. INTRODUCTION

Biomedical research has long shown that human genome contains data from which information about their individual owners, and those related to them can be extracted [51, 63–65]. A lot of privacy-sensitive information are laced all over genomic data, which constitutes enormous worry for individuals whose data are available in electronic format [66–68]. The benefits of continuous research involving the genomic data are equally rife, these include: preventive and personalised healthcare, patient's response to treatment, predisposition to diseases, identification of new drug targets and perhaps a better understanding of cancer [51, 55, 65, 69–74]. On the other hand, when genomic data is used for research or processed by medical personnel, they become exposed to possible misuse and even loss to unauthorised hands. In the face of this possibility, the risk of re-identifying individuals from an available genomic data calls for serious concern [55, 64–66, 68, 75], and has been recognised as a realistic threat. Other unwanted scenarios which could occur as direct consequence of leaking genomic data include: stigmatisation, discrimination, loss of insurance and even loss of employment opportunities for persons whose genomic data is public [54].

What is more worrisome about misuse of genomic data is the fact that the genome has longevity, when leaked, it can neither be revoked nor modified. So, it is obvious that this piece of data is highly sensitive and requires protection that should be adaptive to future security threats. Hence one can claim that any realistic solution should be one which, the security guarantees of the underlying primitives used for implementation should withstand post quantum attacks. Therefore privacy protection techniques have been proposed as an adaptive solution by the cryptography community. The aim will be to allow productive research that utilise genomic data, while eliminating the privacy-risks inherent around these procedures.

Being that no standalone solution can best fit the challenge posed, it is considered that a good combination of *ethical, legal* and *technological* constraints can be employed, to properly manage the risks of privacy leaks that are otherwise possible within this research domain. Owing to this premise, our work seeks to contribute a technological solution to the underlying problem.

In the era of distributed computing, even the medical field has not been left out. It has been common for researchers and medical personnel to work without boundaries of country borders, albeit, via a virtual collaboration [63, 68, 76]. This means that more data can now be shared for research purposes and even diagnosis of diseases [67]. It also presents us with the possibility of allowing cloud services process medical data, even when they do not reside in the same country as the owners of the data. This need for collaboration, data sharing and cloud processing of genomic data further pushes for privacy-preserving secure computing protocols [63, 76].

Having a genomic dataset and controlling access to it is the main aim of this work. In a nutshell, this means that while these data is not available to the public, experts who need them for research are granted restricted access to only subsets relevant to their work [75]. Such access for processing data may include string searching and comparison, as well as GWAS computations.

Genome Wide Association Studies: As highlighted in [51, 77], the first ever human

genome sequencing was achievable in 2001, after directly gulping a whooping US-$300 million from the initial budget of US-$3 billion. Fast-forward 6 years later, and the same feat is feasible for about US-$100,000. In 2006 [77], it was anticipated that in 2014, a further reduction to US-$1,000 was possible for sequencing the human genome. Recent literature [52, 71] have even suggested that a meagre US-$100, will be a reality in the very near future. If that be the case, one can deduce that amongst other possibilities, a direct consequences of affordable genomic data would be the torrential flow of genomic data *in silico*. It is obviously a good development for researchers, who would heavily rely on these data to improve on their research, refine and optimise diagnosis and many others positive possibilities. With a wealth of data in the form of genomic data lying at the disposal of researchers and medical personnel, learning and inferring from these data becomes an indisputable objective.

Without loss of generality in description, GWAS can simply be simplified to the activities presented above, it is about gathering genetic data, processing them and relying on them to investigate relationship (association) of genes to common known diseases. It will be possible to even detect unknown diseases and the effect of drugs on treatments. With GWAS researchers can now measure, analyse and predict previously unknown genetic influence on a person, this can help in early detection and prevention of certain diseases, as well as personalised healthcare. For useful gene-disease associations to be estimated, some computations become handy, and these will be discussed in subsection 3.3.2. Nonetheless, most of the computations can easily put the data owners at privacy-risk. It has led to the suggestion that protection of genomic data is a necessity, to address possible ethical, political, technological and privacy concerns. From the technological solution approach, we hope to address the privacy-threats using cryptographic primitives. Just to mention, with genomic data, data anonymization is not enough guarantee to avoid re-identification and also, conventional encryption might not offer much better protection against envisaged privacy-threats. These can simply be derived from the fact that the said data have longevity, their importance persists even after the demise of the data owner.

Related Works: Realising the privacy-sensitive nature of genomic data, researchers have delved into search for privacy-preserving solutions, in the hope to protect privacy of owners while still being able to process and compute operations using these data. Some of these works are discussed here. Privacy-preserving GWAS spans across more possibilities than just GWAS-Computations. According to [75], other important categories include:

- Private string searching and comparison.
- Private release of aggregated data.
- Private read mapping.

of course, this list is not in itself exhaustive, but we will only consider works that directly address computations very peculiar to GWAS. As early as 1999 [78–80], some researchers had anticipated privacy risks involved with genomic data. So they proposed denominalization and de-identification as protection schemes, to preserve privacy. This

did not stop re-identification attacks from being hugely successful, as discussed in [55]. Other authors [81] have subsequently recommended *Trusted Third Parties* and *Semi-trusted Third Parties* but then, it is not always easy to completely trust a third party, who could still be susceptible to coercion, compulsion and even corruption to be compromised. More recently in [82], attempts were made to analyse genomic data while avoiding privacy-invasion of participants of the data. Summarily, they adopted differential privacy as a privacy-preserving technique, and documented to have obtained utility with their procedure. However, addition of noise using differential privacy is not a silver bullet to deflate possible re-identification. Especially when the published data can be augmented with other side information. But most importantly is the fact that differential privacy contains noise, which will evidently affect the utility, no matter the degree of noise. This is a huge trade-off, but it is only left for the geneticists and bio-statisticians to decide if the noise only contributes a negligible disturbance to the final results.

While the last paper approach to resolving possible privacy breaches is via differential privacy, [83] chooses to adopt a different approach. The authors adopt homomorphic encryption as a tool to enable analysis of these privacy sensitive data. Homomorphic Encryption holds a lot of promises, and if its capabilities are optimally harnessed, can become a very productive primitive in guaranteeing privacy for processing genomic data. In this work, different scenarios are considered which include a setting that allows outsourcing encrypted genomic data to a cloud service. In the mentioned scenario, operations on the data by the cloud are still possible, without divulging the decryption keys but still hopeful of achieving utility.

Homomorphic Encryption was further relied on by some other team of researchers [56]. A shot was given to providing privacy guarantees on processing of genomic data, only that this time the focus was on homomorphic encryption scheme whose structures rely on RLWE (Ring Learning With Error). [56] documents an efficiency-improvement from existing implementation of GWAS using homomorphic encryption. They showed that χ^2 test for independence was achievable with improvement in both computation and communication time from existing implementations.

Subsequently, another team of researchers went further to demonstrate how much information can be extracted from computation of genomic data, even on the encrypted domain [35]. Basic genomic algorithms which are common to GWAS are shown to be implementable on encrypted genotype and phenotype data. Lauter et al. [35] report results that preserve utility of the original implementation (computation on unencrypted genomic data). Some of the algorithms demonstrated in their work include:

- Estimation Maximization (EM) algorithm for haplotyping.
- The D and r^2-measures of linkage disequilibrium.
- Cochran-Armitage Test for Trend.

Also worth mentioning is the fact that this implementation relied on Homomorphic Encryption with assumption on RLWE.

Scenario and Assumptions: For the sake of this work, we will explicitly spell out the scenario in which our proposed protocol is targeted, and necessary assumptions. Our setting adopts the semi-honest security model, hence we assume that all parties will

correctly follow the protocol by performing the right computations, but with a curiosity to observe the transitions of the protocol with a view to learning more details than they are statutorily allowed to learn. We assume that a researcher *Alice* is interested in a particular computation, say *Minor Allele Frequency (MAF)*. The data source or cloud *Bob*, who happens to have the computational powers not acquired by *Alice*, is trusted to perform all requests by performing the computation on encrypted data. The result of the computation (which however, is also encrypted), is returned to *Alice*.

3.3.2. PRELIMINARIES

Up until here, we have established a clear direction to the challenge we hope to address. A genomic dataset is at our disposal and we intend to preserve privacy of data in the face of effective computations. So, we propose a protocol that encrypts all genomic data and outsources storage of these data to a semi-honest cloud service who possesses the computational requirements to run these expensive computations. It will be pertinent to have a mental picture of typical algorithms that will be deployed to perform computation, and how our cryptographic privacy enhancing technology optimally fits for a solution. Most of the algorithms are statistical operations that are often required by biostatisticians when trying to learn information from a dataset. And just like most statistical equations require simple arithmetic operation at the least, we show that our adopted primitive (homomorphic encryption), does provide us with the capabilities to perform simple *addition*, *multiplication*, and with a little more effort *division*.

GWAS COMPUTATION

Only a few statistical computations that are usually handy in GWAS are presented.

Minor Allele Frequency: Finding the ratio for which an allele of interest that is at a locus, occurs in a particular population of study is the allele frequency. *MAF* is therefore the allele frequency of the least common *allele*, which appears in that population. If we have a gene with two possible alleles say **A** and **S**, then in a monoploid gene setting, the allele frequency *f()* for **A** is simply computed as follow:

$$f(\mathbf{A}) = \frac{\sum_1^n \mathbf{AA}}{\sum_1^n \mathbf{AA} + \sum_1^m \mathbf{SS}} \tag{3.13}$$

where $N = n + m$ is the total population sample, and n and m are the counts of alleles **A** and **S** respectively. That was rather too easy, owing to the fact that we only have two possible genotypes, which are results of pure combination of possible alleles. What happens when we consider diploid gene settings? Using the same alleles at a particular locus, we consider the following expressions: **AA**, **AS** and **SS**. Just like we did above, we shall try to compute the frequency of the allele **A**. Let genotype distribution be as follows:
$\mathbf{A} = 19, \mathbf{AS} = 21, \mathbf{SS} = 07$.

$$f(\mathbf{A}) = \frac{2 * \sum_1^n \mathbf{AA} + \sum_1^k \mathbf{AS}}{2(\sum_1^n \mathbf{AA} + \sum_1^k \mathbf{AS} + \sum_1^m \mathbf{SS})} \tag{3.14}$$

The total genotype count in this case is $N = n+k+m$, where n, k and m are counts for

AA, **AS** and **SS** respectively. To compute the allele frequency of **A** using the values already presented, we will have

$$f(\mathbf{A}) = \frac{2 * \mathbf{AA} + \mathbf{AS}}{2 * (\mathbf{AA} + \mathbf{AS} + \mathbf{SS})} = \frac{2 * 19 + 21}{2 * (19 + 21 + 07)} = \frac{59}{94} = 0.6277 \tag{3.15}$$

Since we only have two possible alleles in this population, the least common allele should be **S**, with MAF of $(1 - 0.6277) = 0.3723$

To calculate the genotype frequencies we have **AA** $= \frac{n}{N}$, **AS** $= \frac{k}{N}$, **SS** $= \frac{m}{N}$

Linkage Disequilibrium: This is the non-random association of alleles at different loci. Unlike the single locus alleles considered previously, we will be considering two loci but mainly retaining the basic statistics we have developed thus far. The aim of this test is to suggest if SNPs at particular loci of interest behave or occur in such a manner that is not believed to be random. So we present two loci with the following alleles: **A**, **a** and **S**, **s**. When two genotype at different loci are independent of each other, Linkage Equilibrium is considered to have occurred. Simply put, this means that Linkage Disequilibrium happens when there is some degree of dependency between the two loci of interest. Leading to the Hardy-Weinberg Equilibrium (HWE), which is said to hold if allele frequencies are preserved in a population across generations, except otherwise altered by an external factor, including evolutional influences. To measure linkage disequilibrium, the following equations are used to compute D and r^2.

$$D = \begin{cases} \frac{f(AS)f(as) - f(As)f(aS)}{min(f(A)f(s), f(a)f(S))} & \text{if } f(AS)f(aa) - f(As)f(aS) > 0 \\ \\ \frac{f(AS)f(as) - f(As)f(aS)}{min(f(A)f(S), f(a)f(s))} & \text{if } f(AS)f(aa) - f(As)f(aS) < 0 \end{cases} \tag{3.16}$$

$$r^2 = \frac{(f(AS)f(as) - f(As)f(aS))^2}{f(A)f(S)f(a)f(s)} \tag{3.17}$$

On the assumption that the allele frequencies can be obtained from encrypted genomic data, then it follows that the above computations can be computed.

HOMOMORPHIC ENCRYPTION

Homomorphic Encryption (HE) is a cryptographic primitive that allows for simple arithmetic operations over a ciphertext space. A HE scheme can either allow for simple addition, multiplication or even both. We have an additive scheme if it can only allow for addition operations and a *fully homomorphic encryption* (FHE) scheme if both addition and multiplication can be harnessed from the scheme. Give two messages m_1 and m_2, an encryption and decryption functions $Enc()$ and $Dec()$ respectively. We have that:

$$Enc(m_1) \oplus Enc(m_2) \rightarrow Enc(m_1 + m_2) : Dec(Enc(m_1 + m_2)) := (m_1 + m_2) \tag{3.18}$$

$$Enc(m_1) \otimes Enc(m_2) \rightarrow Enc(m_1 * m_2) : Dec(Enc(m_1 * m_2)) := (m_1 * m_2) \tag{3.19}$$

In 2009, [17] proposed an FHE scheme, which reduced its security to some well known difficult lattice problem. Further works were done to improve the original scheme, due to the complexity involved in implementation. Bringing about works like [84–86], which have been able to present a levelled homomorphic encryption scheme, that is capable of handling multiplication to a certain degree or depth, before the ciphertext becomes un-decryptable. The main idea is that for every operation, some noise is added to the ciphertext, and when this noise grows above a certain threshold, decryption of the ciphertext becomes a problem. While $addition$ contributes small degree of noise, $multiplication$ allows the noise to grow very fast. These schemes often reduce their security to lattice based problems like *shortest vector problem* (SVP) including *ring learning with error problems* (RLWE). Because the multiplication function obtainable from these homomorphic encryptions are not arbitrary (as to control the noise growth), it is labelled *levelled* or *somewhat homomorphic encryption* (SHE). To show that the multiplication depth can only go as deep as the specified level, during parameter setup.

With a SHE scheme handy, and statistical algorithms available, we can then deploy this primitive to solve the arithmetic operations we identified earlier. It can be demonstrated that with SHE, these algorithms can be computed while preserving the utility and not trading privacy of the genomic data concerned.

3.3.3. PRIVACY PRESERVING χ^2 STATISTIC

In GWAS computation, X^2 is often computed and compared to the χ^2 distribution. A common test can be applied to know if the HWE holds in a given distribution. An example of a computation is presented below:

$$X^2 = \sum_{i=\{AA,AS,SS\}} \frac{(O_i - E_i)^2}{E_i} \tag{3.20}$$

O_i and E_i represent observed frequency allele and Expected frequency allele of the population. Since the frequency allele can easily be computed by simple addition and multiplication, and the required arithmetic operations are obtainable in our discussed Homomorphic Encryption. It can be concluded that the χ^2 statistics can be computed in a privacy-preserving manner, over encrypted dataset. Other computations such as the Cochran-Armitage Test for Trend can equally be computed using this procedure, and even meta-analysis of data from different experiments can be produced as well. For simplicity, we shall show how X^2 test statistic can be computed, borrowing the suggestions in [35, 56, 59], with a subtle modification. Every SNP representation is assumed to belong to a genotype classification. And for a single locus test, we produce 3 encryptions, $Enc(x)_{c,d} : x \in \{0,1\}$, c and d are row and column indexes respectively. The rows depict the SNPs for participants, while the columns depict genotype (AA, AS, SS). Assuming that all loci representation correctly fall into a genotype class, then the summation of the row values $\sum_{c=1}^{N}$ will produce n, k and m, recall that $N = n+k+m$. It then becomes feasible to calculate the sum of the genotypes by simply adding the encrypted values for each column. This will require a constant cost of $3N$ numbers of additions using homomorphic encryption.

$$X^2 = \frac{(n - E_{AA})^2}{E_{AA}} + \frac{(k - E_{AS})^2}{E_{AS}} + \frac{(m - E_{SS})^2}{E_{SS}} \tag{3.21}$$

$$X^2 = \frac{(n - E_{AA})^2 * E_b * E_c + (k - E_{AS})^2 * E_a * E_c + (n - E_{SS})^2 * E_a * E_c}{E_{AA} * E_{AS} * E_{SS}} \tag{3.22}$$

Again, to compute the X^2 test statistic, it becomes evident that this computation will require at least, $(3N + 5)$ *additions, 14 multiplications* and a single $division$. We deliberately ignore the computation of $E_{i=\{AA,AS,SS\}}$, since those can be easily pre-computed and stored. But if we have a (2×2) or (2×3) contingency table as presented in [59], we can still show that these complex looking computations can be reduced to $additions$, $multiplications$, and a single $division$. Since our SHE scheme can perform $addition$ and $multiplication$ efficiently, we are left to show that a trivial non-cryptographically secure means can be used to efficiently carry out the division. We offer this trivial solution, with the knowledge that a cryptographically secure division will involve a multi-party computation, of which we do not wish to discuss, due to the complexities involve. The non-trivial solution would be as follows:

$$\frac{Enc(x)}{Enc(y)}, \quad r \leftarrow \mathbb{R}, \quad \frac{Enc(x) \otimes Enc(r)}{Enc(y) \otimes Enc(r)} \tag{3.23}$$

Both numerator and denominator are presented to the researcher, who can decrypt them and perform the division in clear. The test statistic is therefore obtained and compared to the appropriate $p - value$ that was chosen, with 1 degree of freedom. The obtained result will not lose utility, and yet achieves a privacy guarantee on the semi-honest settings. The cloud to whom data processing is outsourced, does not know what values are encrypted, but can perform operations using only the ciphertext, and the researcher who queries the database for X^2 value can be sure to obtain a correct value.

COMPLEXITY:

The complexity of the proposed protocol can only be as efficient as the HE scheme deployed to solve the problem. For instance, when computing allele frequencies, several additions and a few multiplications are required. Which means that the computational complexity can be bounded by the computational complexity of the underlying HE scheme. However, if an additive HE scheme is to be deployed, we envisage an extra cost associated with communication. This is because multiplication in additive schemes are often performed as a multi-party computation (MPC). For the simple case of computing X^2, we have a cost of $3N+5$ additions, 14 multiplications and 1 division. Which will involve many rounds of communication for an additive homomorphic encryption scheme.

PRIVACY ANALYSIS

The privacy proof of this proposal is inherited from the security of the encryption scheme adopted for the implementation. So long as the homomorphic encryption scheme is proven secure, the proposed protocol should remain privacy-proof.

For future work, we strongly recommend adoption of SHE scheme over an additive HE scheme like *Paillier* . Also, division over encrypted domain can be attempted in order to address the bottleneck of having to transmit division operands in the clear. This should be an important addition to this work, and perhaps one can leverage on that to perform even faster computations of statistical GWAS algorithms.

3.3.4. CONCLUSION

With major enhancement of the described cryptographic primitives, we foresee further deployment of privacy enhancing techniques to create protocols for processing of genomic data. We believe that this is an achievable feat in the near future, as to prepare for the bloat in availability of genomic data *in silico*. This protocol should be able to preserve the utility of results as obtainable in unencrypted data scenario and better than anonymized data implementation. Though the performance values will be expensive as a result of the encrypted data and encoding needed to be done, we believe that with further attention paid to this area of research, performance optimization is very realistic.

REFERENCES

[1] S. Goodwin, J. D. Mcpherson, and W. R. Mccombie, *Coming of age: ten years of next-generation sequencing technologies,* Nature Reviews Genetics **17**, 333–351 (2016).

[2] R. Wu, *Nucleotide sequence analysis of DNA,* Nature New Biology **236**, 198–200 (1972).

[3] F. Sanger, G. M. Air, B. G. Barrell, N. L. Brown, A. R. Coulson, J. C. Fiddes, C. A. Hutchison, P. M. Slocombe, and M. Smith, *Nucleotide sequence of bacteriophage x174 DNA,* Nature **265**, 687–695 (1977).

[4] *About the human genome project,* `https://web.ornl.gov/sci/techresources/Human_Genome/project/index.shtml`, accessed: 2018-03-21.

[5] 23andMe, *23andme,* (2018), `https://www.23andme.com/en-eu/` Online; accessed January, 2018.

[6] B. Malin and L. Sweeney, *How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems,* Journal of biomedical informatics **37**, 179 (2004).

[7] E. W. Clayton, B. J. Evans, J. W. Hazel, and M. A. Rothstein, *The law of genetic privacy: applications, implications, and limitations,* Journal of Law and the Biosciences (2019).

[8] J. E. Lunshof, R. Chadwick, D. B. Vorhaus, and G. M. Church, *From genetic privacy to open consent,* Nature Reviews Genetics **9**, 406 (2008).

[9] M. Naveed, E. Ayday, E. W. Clayton, J. Fellay, C. A. Gunter, J.-P. Hubaux, B. A. Malin, and X. Wang, *Privacy in the genomic era,* ACM Computing Surveys **48**, 1–44 (2015).

[10] E. Ayday, E. D. Cristofaro, J.-P. Hubaux, and G. Tsudik, *Whole genome sequencing: Revolutionary medicine or privacy nightmare?* Computer **48**, 58–66 (2015).

[11] T. Bradley, X. Ding, and G. Tsudik, *Genomic security (lest we forget),* IEEE Security & Privacy **15**, 38–46 (2017).

[12] *What are single nucleotide polymorphisms (snps)? - genetics home reference,* https://ghr.nlm.nih.gov/primer/genomicresearch/snp, accessed: 2018-04-09.

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

[14] R. L. Rivest, L. Adleman, and M. L. Dertouzos, *On data banks and privacy homomorphisms,* Foundations of secure computation **4**, 169 (1978).

[15] P. Paillier, *Public-key cryptosystems based on composite degree residuosity classes,* in *Advances in cryptology?EUROCRYPT?99* (Springer, 1999) pp. 223–238.

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

[17] C. Gentry, *Fully homomorphic encryption using ideal lattices,* in *Proceedings of the forty-first annual ACM symposium on Theory of computing* (2009) pp. 169–178.

[18] D. Micciancio and O. Regev, *Lattice-based cryptography,* https://www.cims.nyu.edu/~regev/papers/pqc.pdf, accessed: 2018-04-09.

[19] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, *Fully homomorphic encryption over the integers,* in *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, 2010) pp. 24–43.

[20] J. H. Cheon, A. Kim, M. Kim, and Y. Song, *Homomorphic encryption for arithmetic of approximate numbers,* in *International Conference on the Theory and Application of Cryptology and Information Security* (Springer, 2017) pp. 409–437.

[21] J. Fan and F. Vercauteren, *Somewhat practical fully homomorphic encryption.* IACR Cryptology ePrint Archive **2012**, 144 (2012).

[22] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, *(leveled) fully homomorphic encryption without bootstrapping,* ACM Transactions on Computation Theory (TOCT) **6**, 1 (2014).

[23] A. Martin, C. Melissa, C. Hao, D. Jintai, G. Shafi, G. Sergey, H. Shai, H. Jeffrey, L. Kim, L. Kristin, L. Satya, M. Daniele, M. Dustin, M. Travis, S. Amit, and V. Vinod, *Homomorphic encryption standard,* (2018), http://homomorphicencryption.org/wp-content/uploads/2018/11/HomomorphicEncryptionStandardv1.1.pdf Online; accessed January, 2020.

[24] C. Dwork, *Differential privacy: A survey of results,* in *International conference on theory and applications of models of computation* (Springer, 2008) pp. 1–19.

[25] C. Dwork, *Differential privacy,* in *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006)*, Vol. 4052 (Springer Verlag, Venice, Italy, 2006) pp. 1–12.

[26] O. Goldreich, *Secure multi-party computation,* Manuscript. Preliminary version , 86 (1998).

[27] R. Cramer, I. B. Damgård, *et al.*, *Secure multiparty computation* (Cambridge University Press, 2015).

[28] M. Bellare, V. T. Hoang, and P. Rogaway, *Foundations of garbled circuits,* in *Proceedings of the 2012 ACM conference on Computer and communications security* (2012) pp. 784–796.

[29] Y. Huang, D. Evans, J. Katz, and L. Malka, *Faster secure two-party computation using garbled circuits.* in *USENIX Security Symposium*, Vol. 201 (2011) pp. 331–335.

[30] M. J. Freedman, K. Nissim, and B. Pinkas, *Efficient private matching and set intersection,* in *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings* (2004) pp. 1–19.

[31] E. D. Cristofaro and G. Tsudik, *Practical private set intersection protocols with linear complexity,* in *Financial Cryptography and Data Security, 14th International Conference, FC 2010, Tenerife, Canary Islands, January 25-28, 2010, Revised Selected Papers* (2010) pp. 143–159.

[32] R. A. Popa, F. H. Li, and N. Zeldovich, *An ideal-security protocol for order-preserving encoding,* in *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013* (2013) pp. 463–477.

[33] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, *Order-preserving encryption for numeric data,* in *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004* (2004) pp. 563–574.

[34] E. Ayday, J. L. Raisaro, U. Hengartner, A. Molyneaux, and J. Hubaux, *Privacy-preserving processing of raw genomic data,* in *Data Privacy Management and Autonomous Spontaneous Security - 8th International Workshop, DPM 2013, and 6th International Workshop, SETOP 2013, Egham, UK, September 12-13, 2013, Revised Selected Papers* (2013) pp. 133–147.

[35] K. Lauter, A. López-Alt, and M. Naehrig, *Private computation on encrypted genomic data,* in *Progress in Cryptology-LATINCRYPT 2014* (Springer, 2014) pp. 3–27.

[36] A. Bouti and J. Keller, *Secure genomic data evaluation in cloud environments,* in *2017 International Symposium on Networks, Computers and Communications, ISNCC 2017, Marrakech, Morocco, May 16-18, 2017* (2017) pp. 1–6.

[37] S. Simmons and B. Berger, *Realizing privacy preserving genome-wide association studies,* Bioinformatics **32**, 1293 (2016).

[38] E. D. Cristofaro, S. Faber, and G. Tsudik, *Secure genomic testing with size- and position-hiding private substring matching,* in *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013* (2013) pp. 107–118.

[39] B. Wang, W. Song, W. Lou, and Y. T. Hou, *Privacy-preserving pattern matching over encrypted genetic data in cloud computing,* in *2017 IEEE Conference on Computer Communications, INFOCOM 2017, Atlanta, GA, USA, May 1-4, 2017* (2017) pp. 1–9.

[40] P. Baldi, R. Baronio, E. D. 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, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011* (2011) pp. 691–702.

[41] X. S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, and D. Bu, *Efficient genome-wide, privacy-preserving similar patient query based on private edit distance,* in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015* (2015) pp. 492–503.

[42] E. D. Cristofaro, S. Faber, P. Gasti, and G. Tsudik, *Genodroid: are privacy-preserving genomic tests ready for prime time?* in *Proceedings of the 11th annual ACM Workshop on Privacy in the Electronic Society, WPES 2012, Raleigh, NC, USA, October 15, 2012* (2012) pp. 97–108.

[43] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. S. Gaur, M. Conti, and M. Rajarajan, *Android security: A survey of issues, malware penetration, and defenses,* IEEE Communications Surveys and Tutorials **17**, 998 (2015).

[44] M. Z. Hasan, M. S. R. Mahdi, and N. Mohammed, *Secure count query on encrypted genomic data,* CoRR **abs/1703.01534** (2017), arXiv:1703.01534 .

[45] M. Namazi, J. R. Troncoso-Pastoriza, and F. Pérez-González, *Dynamic privacy-preserving genomic susceptibility testing,* Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security - IH&MMSec 16 , 45–50 (2016).

[46] G. Danezis and E. D. Cristofaro, *Fast and private genomic testing for disease susceptibility,* Proceedings of the 13th Workshop on Privacy in the Electronic Society - WPES 14 , 31–34 (2014).

[47] E. Ayday, J. L. Raisaro, J.-P. Hubaux, and J. Rougemont, *Protecting and evaluating genomic privacy in medical tests and personalized medicine,* Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society - WPES 13 (2013), 10.1145/2517840.2517843.

[48] X. Lei, X. Zhu, H. Chi, and S. Jiang, *Cloud-assisted privacy-preserving genetic paternity test,* 2015 IEEE/CIC International Conference on Communications in China (ICCC) (2015), 10.1109/iccchina.2015.7448655.

[49] Y. Yamamoto and M. Oguchi, *A decentralized system of genome secret search implemented with fully homomorphic encryption,* 2017 IEEE International Conference on Smart Computing (SMARTCOMP) (2017), 10.1109/smartcomp.2017.7946977.

[50] H. Perl, Y. Mohammed, M. Brenner, and M. Smith, *Fast confidential search for biomedical data using bloom filters and homomorphic cryptography,* 2012 IEEE 8th International Conference on E-Science , 1–8 (2012).

[51] E. S. Lander, L. M. Linton, B. Birren, C. Nusbaum, M. C. Zody, J. Baldwin, K. Devon, K. Dewar, M. Doyle, W. FitzHugh, *et al.*, *Initial sequencing and analysis of the human genome,* Nature **409**, 860 (2001).

[52] D. McMorrow, *The $100 genome: Implications for the DoD*, Tech. Rep. (DTIC Document, 2010).

[53] S. Wang, X. Jiang, D. Fox, and L. Ohno-Machado, *Preserving genome privacy in research studies,* in *Medical Data Privacy Handbook* (Springer, 2015) pp. 425–441.

[54] Z. Lin, A. B. Owen, and R. B. Altman, *Genomic research and human subject privacy,* SCIENCE-NEW YORK THEN WASHINGTON-. , 183 (2004).

[55] B. A. Malin, *An evaluation of the current state of genomic data privacy protection technology and a roadmap for the future,* Journal of the American Medical Informatics Association **12**, 28 (2005).

[56] W. Lu, Y. Yamada, and J. Sakuma, *Efficient secure outsourcing of genome-wide association studies,* in *Security and Privacy Workshops (SPW), 2015 IEEE* (IEEE, 2015) pp. 3–6.

[57] A. Ziegler, F. Pahlke, *et al.*, *A Statistical Approach to Genetic Epidemiology: Concepts and Applications, with an E-learning platform* (John Wiley & Sons, 2010).

[58] B. S. Weir, *Inferences about linkage disequilibrium,* Biometrics , 235 (1979).

[59] L. Kamm, D. Bogdanov, S. Laur, and J. Vilo, *A new way to protect privacy in large-scale genome-wide association studies,* Bioinformatics **29**, 886 (2013).

[60] D. Wu and J. Haven, *Using homomorphic encryption for large scale statistical analysis,* (2012).

[61] A. Shahbazi, F. Bayatbabolghani, and M. Blanton, *Private computation with genomic data for genome-wide association and linkage studies,* in *Proc. 3rd Int. Workshop Genome Privacy Security* (2016).

[62] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, *Generating private recommendations efficiently using homomorphic encryption and data packing,* Information Forensics and Security, IEEE Transactions on **7**, 1053 (2012).

[63] M. M. Baig, J. Li, J. Liu, H. Wang, and J. Wang, *Privacy protection for genomic data: current techniques and challenges* (Springer, 2010).

[64] S. Jha, L. Kruger, and V. Shmatikov, *Towards practical privacy for genomic computation,* in *Security and Privacy, 2008. SP 2008. IEEE Symposium on* (IEEE, 2008) pp. 216–230.

[65] F. Bruekers, S. Katzenbeisser, K. Kursawe, and P. Tuyls, *Privacy-preserving matching of dna profiles.* IACR Cryptology ePrint Archive **2008**, 203 (2008).

[66] C. J. Willer, Y. Li, and G. R. Abecasis, *Metal: fast and efficient meta-analysis of genomewide association scans,* Bioinformatics **26**, 2190 (2010).

[67] 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 , btu561 (2014).

[68] J. Wagner, J. N. Paulson, X.-S. Wang, B. Bhattacharjee, and H. C. Bravo, *Privacy-preserving microbiome analysis using secure computation,* bioRxiv , 025999 (2015).

[69] C. Cao and J. Moult, *GWAS and drug targets,* BMC Genomics. , 15(Suppl 4):S5 (2014).

[70] J. Zhang, K. Jiang, L. Lv, H. Wang, Z. Shen, Z. Gao, B. Wang, Y. Yang, Y. Ye, and S. Wang, *Use of genome-wide association studies for cancer research and drug repositioning,* PloS one **10**, e0116477 (2015).

[71] D. L. Selwood, *Beyond the hundred dollar genome–drug discovery futures,* Chemical biology & drug design **81**, 1 (2013).

[72] L. A. Hindorff, P. Sethupathy, H. A. Junkins, E. M. Ramos, J. P. Mehta, F. S. Collins, and T. A. Manolio, *Potential etiologic and functional implications of genome-wide association loci for human diseases and traits,* Proceedings of the National Academy of Sciences **106**, 9362 (2009).

[73] F. Liu, A. Arias-Vásquez, K. Sleegers, Y. S. Aulchenko, M. Kayser, P. Sanchez-Juan, B.-J. Feng, A. M. Bertoli-Avella, J. van Swieten, T. I. Axenovich, *et al.*, *A genomewide screen for late-onset alzheimer disease in a genetically isolated dutch population,* The American Journal of Human Genetics **81**, 17 (2007).

[74] C. C. Spencer, Z. Su, P. Donnelly, and J. Marchini, *Designing genome-wide association studies: sample size, power, imputation, and the choice of genotyping chip,* PLoS Genet **5**, e1000477 (2009).

[75] E. Ayday, M. Humbert, J. Fellay, M. Laren, P. Jack, J. Rougemont, J. L. Raisaro, A. Telenti, and J.-P. Hubaux, *Protecting personal genome privacy: Solutions from information security,* Tech. Rep. (EPFL, 2012).

[76] M. Kantarcioglu, W. Jiang, Y. Liu, and B. Malin, *A cryptographic approach to securely share and query genomic sequences,* Information Technology in Biomedicine, IEEE Transactions on **12**, 606 (2008).

[77] G. Church, *The race for the $1000 genome,* Science **311** (2006).

[78] D. Gaudet, S. Arsenault, C. Bélanger, T. Hudson, P. Perron, M. Bernard, and P. Hamet, *Procedure to protect confidentiality of familial data in community genetics and genomic research,* Clinical genetics **55**, 259 (1999).

[79] L. Burnett, K. Barlow-Stewart, A. Proos, and H. Aizenberg, *The" genetrustee": a universal identification system that ensures privacy and confidentiality for human genetic databases.* Journal of Law and Medicine **10**, 506 (2003).

[80] J. E. Wylie and G. P. Mineau, *Biomedical databases: protecting privacy and promoting research,* Trends in biotechnology **21**, 113 (2003).

[81] G. De Moor, B. Claerhout, F. De Meyer, *et al.*, *Privacy enhancing techniques the key to secure communication and management of clinical and genomic data,* Methods Archive **42**, 148 (2003).

[82] 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* (ACM, 2013) pp. 1079–1087.

[83] J. W. Bos, K. Lauter, and M. Naehrig, *Private predictive analysis on encrypted medical data,* Journal of biomedical informatics **50**, 234 (2014).

[84] Z. Brakerski and V. Vaikuntanathan, *Fully homomorphic encryption from ring-lwe and security for key dependent messages,* in *Annual cryptology conference* (Springer, 2011) pp. 505–524.

[85] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, *(leveled) fully homomorphic encryption without bootstrapping,* in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (ACM, 2012) pp. 309–325.

[86] M. Yagisawa, *Fully homomorphic encryption without bootstrapping.* IACR Cryptology ePrint Archive **2015**, 474 (2015).

4

POST-GWAS COMPUTATION ON PRIVACY-SENSITIVE DATA

Privacy-Preserving genome processing as a service is a viable business model that is beneficial to both the investor and the teaming users seeking to enjoy the benefits of genome predictive medicine using genome data without the fear of privacy scandals.

Parts of this chapter have been published as:
(1) Ugwuoke, C., Erkin, Z., Reinders, M., & Lagendijk, R. (2020, March). PREDICT: Efficient Private Disease Susceptibility Testing in Direct-to-Consumer Model. *In Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy (pp. 329-340)*.

4.1. EFFICIENT DST WITH PRIVACY GUARANTEE

Genome sequencing has rapidly advanced in the last decade, making it easier for anyone to obtain digital genomes at low costs from companies such as Helix, MyHeritage, and 23andMe. Companies now offer their services in a direct-to-consumer (DTC) model without the intervention of a medical institution. Thereby, providing people with direct services for paternity testing, ancestry testing and disease susceptibility testing (DST) to infer diseases' predisposition. Genome analyses are partly motivated by curiosity and people often want to partake without fear of privacy invasion. Existing privacy protection solutions for DST adopt cryptographic techniques to protect the genome of a patient from the party responsible for computing the analysis. Said techniques include homomorphic encryption, which can be computationally expensive and could take minutes for only a few single-nucleotide polymorphisms (SNPs). A predominant approach is a solution that computes DST over encrypted data, but the design depends on a medical unit and exposes test results of patients to the medical unit, making the design uncomfortable for privacy-aware individuals. Hence it is pertinent to have an efficient privacy-preserving DST solution with a DTC service. We propose a novel DTC model that protects the privacy of SNPs and prevents leakage of test results to any other party save for the genome owner. Conversely, we protect the privacy of the algorithms or trade secrets used by the genome analyzing companies. Our work utilizes a secure obfuscation technique in computing DST, eliminating expensive computations over encrypted data. Our approach significantly outperforms existing state-of-the-art solutions in runtime and scales linearly for equivalent levels of security. As an example, computing DST for 10,000 SNPs requires approximately 96 milliseconds on commodity hardware. With this efficient and privacy-preserving solution which is also simulation-based secure, we open possibilities for performing genome analyses on collectively shared data resources.

4.2. INTRODUCTION

The technology for sequencing the human genome continues to improve [1, 2], just as the quest to substantially reduce the cost of sequencing has drawn enormous attention to the medical field, research, and commercial platforms in the last two decades [3–7]. Consequently, research publications corroborate the plummeting in time required for obtaining a digital version of the human genome[2, 8–11]. For as low as $100, commercial companies such as Helix, MyHeritage and 23andMe state that they are able to unlock your genome and deliver health and ancestry services directly to customers [12–14], without intervention of your medical doctor. The downward trajectory of the cost and time for sequencing over the last two decades suggests an inevitable upsurge in availability of digital genome in the near future. Furthermore, digital genome is required by the research community for various studies that help in enhancing our understanding of the basic building blocks of life, the relationship between a gene and a phenotype, better understanding of diseases and their causes, patient responses to treatment, and preventive medicare. Only recently, the drug giant GlaxoSmithKline announced that DNA results from the 5 million customer base of 23andMe will be used to design new drugs, and GlaxoSmithKline have also invested $300 million in 23andMe who are already valued at about $1.75 billion [15, 16]. Commercial platforms have also taken advantage by commodification of the genome, and thus, services such as sequencing, ancestry testing and disease susceptibility testing are now offered to willing customers.

Nucleotides are the building blocks for deoxyribonucleic acid (DNA), which can take any of 4 possible bases (A, T, C, G). Often, there is a genetic variation between individuals of the same species, and this variation could happen as a result of a single substitution of a nucleotide base. A single variation in the nucleotide is known as a single-nucleotide polymorphisms (SNP) if it is not observed in more than 1% of the population [17]. Genome Sequencing is usually the first services a customer procures from a commercial platform, allowing an *in vitro* sample of the customer to be used in obtaining an *in silico* dataset. The *in silico* data is commonly presented in the form of SNPs. The National Library of Medicine records that there are roughly 10 million SNPs in the human genome, and more SNPs are still being identified [17]. SNPs have been identified to contribute to an individual's susceptibility to certain diseases [18–22]. After obtaining a customer's digital SNPs, the commercial platform can further analyze the dataset for various reasons or services, common of which is the customer's predisposition to diseases.

Our work is interested in how commercial platforms utilize SNPs in providing DST services to its customers within a direct-to-consumer market model. For instance, the conventional process of conducting a DST using the SNPs of an individual is comprised of four basic phases. In the first phase, a sequencer who is a commercial platform sequences the biological genome and obtains a digital version for storage and further analysis. The Second phase consists of a genome owner, to whom a digital genome has been provided, requesting a disease susceptibility test on a particular disease of interest. In the third phase, upon receiving such request, a commercial platform will request SNPs it deems relevant for such a test, and will perform the DST with the SNPs provided by the customer. Finally, the fourth phase is when the DST results are handed over to the genome owner or his doctor. This setting whereby commercial platforms render ser-

vices such as analyzing the genome, directly to the customers qualifies as a direct-to-consumer (DTC) model [23, 24]. Some customers will use these services to settle paternity or maternity disputes, others will use it to trace their ancestry, or to inspect their genome for disease associated variants [23, 25, 26]. The paradigm shift towards a direct-to-consumer model will impact our medical knowledge as an overwhelming amount of digital genomes will become available as these services become popular [27]. In fact, a survey conducted by Lewis [28] and reported by Pascal [23], documents that as much as 94% of people choose genetic testing out of curiosity.

This paradigm shift towards a DTC service delivery, however, requires that one protects the privacy and security of customers' data when being shared with an untrusted third party. Su and Mclaren et al. [23, 29] argue that the DTC requires a robust combination of regulatory and legal solutions, in order to preserve the confidence that consumers have in using the solution. The implication is that we need efficient privacy-preserving methods for computing on genome data in other to satisfy the customer. In the current model as practiced by the commercial platform, the SNPs of the customer are transmitted *in clear*, which means that at least the commercial platform presents an immediate privacy-risk to the customer. It is difficult to prove that a commercial platform will always adhere to the rules and follow ethical guidelines in protecting the privacy of the genome data, as coercion and disgruntled employees could circumvent such trust models. In fact, only a few 100s SNPs is already enough to threaten the privacy of the customer, by re-identifying an individual even in a large dataset of genetic data [30]. This is further complicated because of familiar relationships between individuals, i.e. information on the genome of a relative also releases information about a customer's own genome. This holds even long after an individual is deceased, thereby posing direct privacy-risks to the relatives. Additionally, whenever genome data is leaked, it is irrevocable and the individual cannot replace the leaked genome with a new set [31]. Together, this places strong privacy requirements for genomic data all through its digital lifespan. Although these customers are only interested in analyzing their genome data for the sheer sake of curiosity [23, 32], it still remains necessary that privacy be guaranteed while satisfying their desire.

The need for providing privacy for all sorts of activities regarding the genome is one that is multifaceted [4], and does require conscious efforts and dedication from legal, ethical, information security and other related research fields. The objective is for customers to have full control over their privacy which requires that genome information about an individual is not shared in a disclosed form with any third party. Secondly, since companies invest a lot of money to understand the genome [16], the SNPs that are relevant to disease and the algorithm for computing DST is regarded as trade secret and should be protected as such. Customers and companies therefore require efficient, provable security and privacy measures that will protect the genome data of a customer and the trade secret of the commercial platform while the customer continues to enjoy services offered by the service provider in the popular DTC model.

Existing information security based privacy-preserving approaches for computing DST commonly adopt cryptographic techniques in plugging the privacy challenges that arises from interactions between the commercial platforms and their customers. For the purpose of simplicity, information security researchers typically concentrate on the last

three phases of the described process. This is important because the sequencing phase is dependent on biological samples which cannot easily be protected with information security techniques. Let us assume that in the first phase, the sequencer deletes all data, both biological and digital relating to the customer, or perhaps the customer now has a secure stand alone device for sequencing the genome. This assumption is consistent with existing solutions [33, 34]. The cryptographic techniques commonly deployed in the last three phases are homomorphic encryption and secret sharing. Homomorphic encryption is a technique that allows anyone to encrypt values in a special way such that basic operations like addition and multiplication can be performed on the encrypted data without using the decryption keys [35–40]. However, computing the DST algorithm over encrypted data is a non-trivial task. Data expansion and computational complexity of homomorphic operations make it expensive, inefficient and hence undesirable for deploying in the wild, as the whole processes is highly time consuming [41, 42]. While secret sharing recommendations are relatively more efficient than their homomorphic encryption counterparts, secret sharing requires that the data be shared amongst various parties with non-collusion restrictions. This does not exactly reflect the ideal scenario as obtainable by current structure of commercial platforms. This is the case because companies do not usually collaborate to compute disease predisposition for a customer.

In this paper, we recommend an information security solution that protects the privacy of customers' data, companies' trade secret and equally preserve the direct-to-consumer market model most profitable for the commercial platforms. Our protocol enhances the efficiency of the runtime by replacing the homomorphic encryption construction with a lightweight obfuscation technique which is provably secure. We provide customers the ultimate power to decide how, when and with whom they choose to share their genome data.

Our contributions are as follows: 1) We propose PREDICT, a novel protocol which executes the existing susceptibility testing requirements with the use of SNPs, and preserves the increasingly popular DTC model adopted by commercial platforms. 2) PREDICT prioritizes the privacy of the customer and that of the commercial platform. Genome data are resident with the customer in a secure format and not stored in a centralized cloud nor shared with any third-party in an unprotected manner. The result of a test can only be deduced by the customer and he is left with the prerogative to either share the result or not. 3) Our protocol is efficient and can be deployed for practical use. Our design and implementation of this privacy-preserving DST protocol significantly outperforms existing privacy-protection solutions in memory and computational efficiency. As an example, it takes about 96 *milliseconds* to compute DST using 10,000 SNPs on a commodity hardware. And finally, we provide privacy proof based on simulation paradigm, as well as the complexity analysis of our protocol.

The outline for the rest of the paper is as follows: in Section 4.3 we discuss relevant literature to our work, and how they differ from our proposal. In Section 4.4 we introduce important building blocks requisite for the construction of our proposed protocol. In Section B.3 we introduce and discuss PREDICT. We present the complexity analysis of PREDICT in Section B.4, followed by optimisation in Section 4.7. We provide further discussion on PREDICT in Section 4.8 and implementation in Section B.5. We present the privacy and security analyses in Section B.6. Finally, in Section B.7, we conclude this

paper.

4.3. RELATED WORK

In this section, we focus on privacy solutions for disease susceptibility testing from an information security perspective. We continue the rest of the discussion with the assumption that every customer or patient already has a digital genome, and the sequencer will play no further role in the interactions. Such digital genome can be securely stored in the cloud or privately kept by the owner in a secure device, this is consistent with proposals in [33, 34]. Previously, a number of other works [23, 29, 33, 34, 43] have proposed solu-

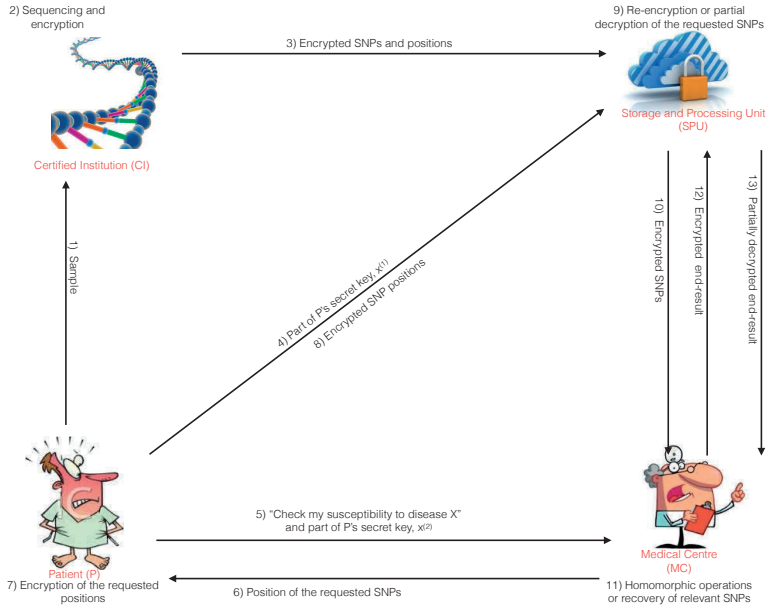

Figure 4.1: Protocol proposed by Ayday et al.[34]

tions for privacy-preserving protocols which perform DST using SNPs or other sensitive data.

A first drawback across these proposals is the inefficiency of the solutions, because adoption of homomorphic encryption introduces significant computational and storage overhead when compared to the non-privacy-preserving solution. One renowned homomorphic encryption based solution is the method proposed by [34]. In the proposal, the bulk of the steps are computations carried out on encrypted data, as seen in Figure 4.1. Following the same protocol as [34], Namazi et al.[43] proposed a similar solution but replaced the homomorphic scheme used by Ayday et al.[34] with an even more computationally expensive homomorphic scheme. This was done to improve privacy and eliminate communication cost peculiar to additive homomorphic schemes. Lastly, Danezis and Cristofaro [33] recommend an improved proposal that is more efficient than the original work by Ayday et al. Although they offer significant improvements in effi-

ciency, their solution equally involves computation over encrypted data. Danezis and Cristofaro also provides a secret-sharing based variant for privacy-preserving DST, but it still uses encrypted data for its computation. Homomorphic encryption techniques are still evolving and techniques to reduce its computational overhead is still an open problem [44, 45]. It implies that homomorphic encryption based approaches are not yet efficient for scalable practical deployment, especially when large dataset are used.

A second drawback to the homomorphic encryption based protocol by Ayday et al.[34] is that of strict privacy. The disease for which a customer is testing can be easily learned by the processing unit, see Figure 4.1. In testing for a disease predisposition, the processing unit is allowed to learn the SNPs that are relevant for the computation. Even though the exact values for the SNPs are not known by the processing unit, this is still a privacy concern. Also, in [34, 43], the final result of the test is transmitted to the medical unit rather than the customer, which is not consistent with our objective of granting the customer absolute control to privacy.

A third concern inherent in the existing works [34, 43] is the fact that the protocols proposed do not seamlessly fit into the model as currently observed between the commercial platforms and their customers. The direct-to-consumer relationship preferred by the commercial platforms is not well reflected in the mentioned proposal. The medical units continue to play an inalienable role in those proposals, which makes these protocol not suitable for a DTC service delivery.

Other drawbacks of the state-of-the-art proposal by Ayday et al.[34] include the following:

- Their approach proposes to store the protected genome data with a storage and processing unit (SPU). This requires that individuals must store their genome data encrypted on a central cloud infrastructure, making it enticing for attackers.

- The protocol assumes that the homomorphic operations are computed at the Medical Centres (MC). This is not practical in the wild as such operations are computationally expensive for average medical centres to carry out. Furthermore, the MC is not often equipped with the technical and security requirements for handling such operations.

We propose a protocol that aids customers and commercial platforms to interact and compute disease susceptibility test in a privacy-preserving manner without being bedeviled with the above listed drawbacks.

4.4. PRELIMINARIES

In this section, we provide the necessary building blocks required to understand our proposed protocol. These include: the cryptographic protocols such as multi-party computation, obfuscation techniques as well as the functions required to compute disease susceptibility testing. In this work, we adopt the *semi-honest* a.k.a *honest-but-curious* security model, which implies that every stakeholder is expected to judiciously follow the rules of the protocol, but can be passively curious to learn extra information from data they can observe.

4.4.1. SECURE MULTI-PARTY COMPUTATION

A secure multi-party computation (MPC) is an interactive cryptographic protocol that allows for two or more mistrusting parties to jointly compute a function using their private data as input [46, 47]. It allows for the output of the desired function to be public but the contributed inputs remains private upon the assumption that each party does not digress from the rules of the protocol. We deploy the concept of MPC by allowing three mis-trusting parties (customers, commercial platforms and processing unit) to jointly compute a DST function using their private inputs. MPCs are commonly designed in a semi-honest security model, because the adversary is considered to be able to control some parties in the protocol.

4.4.2. HOMOMORPHIC ENCRYPTION

Homomorphic encryption allows for arbitrary algebraic operations to be performed on ciphertexts. Let $Enc_{pk}(\cdot)$ and $Dec_{sk}(\cdot)$ represent encryption and decryption functions respectively. (m_1, m_2) are two messages and k is a scalar value, while $\boxplus$, $\boxdot$ and $\boxtimes$ are arbitrary operations on the ciphertexts. Then, homomorphism is defined as:

$$Dec_{sk}(Enc_{pk}(m_1) \boxplus Enc_{pk}(m_2)) = m_1 + m_2\,, \tag{4.1}$$

$$Dec_{sk}(Enc_{pk}(m_1) \boxdot Enc_{pk}(m_2)) = m_1 \cdot m_2\,, \tag{4.2}$$

$$Dec_{sk}(Enc_{pk}(m_1) \boxtimes k) = m_1 \cdot k\,. \tag{4.3}$$

PAILLIER SCHEME [37] :

Paillier cryptosystem is an additively homomorphic scheme. Given a public key, private key pair (pk, sk), and $n := p \cdot q$, s.t p and q are distinct large primes, $pk := (g, n)$ and $sk := \lambda(n)$, where g is generator of order n, $\lambda(n)$ is the Carmichael's function on n, expressed as $\lambda(n) := lcm(p-1, q-1)$.

Enc: $c := Enc_{pk}(m, r) := g^m \cdot r^n \ mod\ n^2$, where $c \in \mathbb{Z}^*_{n^2}$; $r \leftarrow \mathbb{Z}^*_n$.

Dec: Given c, $m := \frac{\mathbb{L}_n(c^\lambda \ mod\ n^2)}{\mathbb{L}_n(g^\lambda \ mod\ n^2)} \ mod\ n$ and $\mathbb{L}_n(a) := \frac{a-1}{n}$.

Additive Homomorphism: Given two ciphertexts of messages $m_0\ and\ m_1$, we can compute the sum as follows:

$$\begin{aligned} Enc_{pk}(m_0, r_0) \times Enc_{pk}(m_1, r_1) &:= (g^{m_0} \cdot r_0^n \times g^{m_1} \cdot r_1^n) \\ &:= (g^{m_0+m_1} \cdot (r_0 \cdot r_1)^n \ mod\ n^2) \\ &:= Enc_{pk}(m_0 + m_1)\,. \end{aligned}$$

With the above mentioned homomorphic scheme, it is possible to compute simple linear functions such as aggregation of encrypted values. However, more complex function that involves division, multiplication and other complex operations on encrypted data are not feasible. Complex functions are usually computed by the introduction of a third party who decrypts ciphertexts, computes the complex operation *in clear* and re-encrypts the results.

It is evident by inspection that ciphertexts are generated modulo n^2 and this results to data expansion for every encrypted value. Computing on the ciphertext introduces

computational overhead since n should not be less than 2048 bits in order to obtain 112-bit security, being that 112-bit security is considered sufficient between the in the year 2016 up until 2030 [48]. The reader is referred to [**?**] for further details on the Paillier scheme.

4.4.3. PRIVACY-PRESERVING DST

Every individual inherits a pair of allele to make a gene at a locus, each allele comes from each contributing parents. An allele can either be major or minor, depending on what percentage of the population have them. Since a gene is made up of two alleles, it can occur in any of the three possible classes: two major alleles, two minor alleles, or a major and minor allele. Subsequently, we use the term 'SNP value' to represent the class in which an individual's SNP falls within. We denote the 'SNP values' as {0, 1, 2} to denote No SNP (two major alleles), heterozygous SNP (a major and minor allele) and homozygous SNP (two minor alleles) respectively. For a DNA sequence belonging to a customer, we denote the SNP at locus i as SNP_i , where $SNP_i \in \{0,1,2\}$.

A *weighting averaging* function is used to compute the susceptibility of a patient to a disease X. A DST can be computed by weighing the contribution of each SNP to the disease X , as follows:

a) Let $L(x)$ represent a set of all known SNPs that contribute to the disease X and C_i represents the weight that a SNP at locus i contributes to the disease X.

b) $Pr(X|SNP_i)$ denotes the probability that an individual has a disease X, conditioned on the SNP value at the locus i.

c) A SNP at locus i contributes $C_i \cdot Pr(X|SNP_i = j)$, with $j \in \{0,1,2\}$.

d) The aggregation of all loci is then $\sum_{i \in L(x)} C_i \cdot Pr(X|SNP_i = j)$.

e) The aggregation is then normalized over all weights to obtain a disease susceptibility test score:

$$S^X = \frac{1}{\sum_{t \in L(x)} C_t} \cdot \sum_{i \in L(x)} C_i \cdot Pr(X|SNP_i = j) . \tag{4.4}$$

It can be seen from Eq. 4.4 that three inputs are required to compute S^X , all of which are considered privacy-sensitive and should not be shared unprotected.

- Only the customer knows SNP_i values, as this is his private data.
- The commercial platform knows C_i and $Pr(X|SNP_i = j)$ values, these being his trade secrets.

By adopting homomorphic encryption to compute Eq. 4.4 in a privacy-protected setting, Ayday et al. has re-written the equation to:

$$S^X = \frac{1}{\sum_{t\in L(x)} C_t} \times \sum_{i\in L(x)} C_i \Big[\frac{p_0^i(X)}{(0-1)(0-2)}[SNP_i - 1]\times$$
$$[SNP_i - 2] + \frac{p_1^i(X)}{(1-0)(1-2)}[SNP_i - 0] \times [SNP_i - 2]$$
$$+ \frac{p_2^i(X)}{(2-0)(2-1)}[SNP_i - 0] \times [SNP_i - 1]\Big], \tag{4.5}$$

where $p_0^i(X) = Pr(X|SNP_i = 0)$, $p_1^i(X) = Pr(X|SNP_i = 1)$ and $p_2^i(X) = Pr(X|SNP_i = 2)$.

In the proposal by Ayday et al. [34], refer to Figure 4.1, Eq. 4.5 is computed homomorphically with the use of an additive homomorphic encryption scheme. Due to the inability of the adopted homomorphic encryption scheme to perform a homomorphic multiplication operation, their protocol is designed to store the encrypted values of $(SNP_i)^2$ which requires additional storage space. The final result as obtained in the protocol described by Ayday et al.[34] is decrypted by the medical unit, who is able to view the result and communicate professional opinion to the patient. However, the storage and processing unit does not have a view of the SNP values *in clear*, despite having to store and process the data. Nevertheless, the storage and processing unit knows which loci have no SNP from those that have at least a single SNP. Therefore the patient's SNPs are not completely private against the storage and processing unit.

Due to the privacy concerns mentioned above and the computational inefficiency introduced by homomorphically computing Eq. 4.4, we introduce a novel protocol for computing DST. Our protocol will optimally compute DST by replacing homomorphic encryption with a secure obfuscation technique using MPC, where each sensitive data input is masked using a one-time-only secure random number. Our proposed protocol removes the medical unit from the setting, replacing it with a commercial platform and ensures that the processing unit does not learn the SNP loci of a customer as was the case in the protocol by Ayday et al. Lastly, in order to achieve these, we again re-write Eq. 4.4, by drawing insight from Eq. 4.5:

a.) Redefine C_i to be the normalized term $\frac{C_i}{\sum_{t\in L(x)} C_t}$.

b.) Re-write Eq. 4.4 to a simpler form using Eq. 4.5,

$$S^X = \sum_{i\in L(x)} C_i \Big[\frac{1}{2} p_0^i (SNP_i - 1)(SNP_i - 2)$$
$$- p_1^i (SNP_i - 0)(SNP_i - 2)$$
$$+ \frac{1}{2} p_2^i (SNP_i - 0)(SNP_i - 1)\Big].$$

c.) Collect like terms,

$$S^X = \sum_{i \in L(x)} C_i \left[SNP_i^2 (\frac{p_0^i}{2} - p_1^i + \frac{p_2^i}{2}) - SNP_i (\frac{3p_0^i}{2} - 2p_1^i + \frac{p_2^i}{2}) + p_0^i \right]. \quad (4.6)$$

From Eq. 4.6, customers own variables SNP_i and SNP_i^2, while the commercial platform owns

$$a_i = C_i (\frac{p_0^i}{2} - p_1^i + \frac{p_2^i}{2}),$$
$$b_i = C_i (\frac{3p_0^i}{2} - 2p_1^i + \frac{p_2^i}{2}),$$
$$v_i = p_0^i.$$

Hence, we can now compute DST as:

$$S^X = \sum_{i \in L(x)} a_i \cdot SNP_i^2 - b_i \cdot SNP_i + v_i. \quad (4.7)$$

4.4.4. SECURE INNER PRODUCT WITH OBFUSCATION

Henceforth, we represent vectors in bold characters, example $\mathbf{X}$, and $|\mathbf{X}|$ denotes the number of elements in $\mathbf{X}$. We will occasionally use the notation $\mathbf{X}[i]$ to represent the i-th term of the vector $\mathbf{X}$. Consider two parties $Alice$ and Bob, each having a vector of values and they wish to compute the inner product of the vectors without revealing the individual values of each vector. $Alice$ holds the vector $\mathbf{X} = \{x_0, x_1, \ldots, x_{n-1}\}$ and Bob holds $\mathbf{Y} = \{y_0, y_1, \ldots, y_{n-1}\}$ both of size $n \in \mathbb{Z}$. They wish to compute the inner product of their vectors $\mathbf{X} \cdot \mathbf{Y} = \sum_{k=0}^{n-1} x_k \cdot y_k$, and the result known only to $Alice$.

In order to solve the secure inner product problem, Du and Atallah[49] propose a three party protocol which uses additive masking to obfuscate the values. Their protocol introduces an untrusted third party $Charlie$ that only helps in data computation. $Charlie$ can be viewed as a cloud infrastructure that helps with the computation of the inner product function. Their protocol is described as follows:

1. $Alice$ and Bob jointly generate two random numbers r and r'.
2. $Alice$ and Bob jointly generate two random vectors $\mathbf{R}, \mathbf{R}'$ of size n.
3. $Alice$ sends $\mathbf{w}_1 = \mathbf{X} + \mathbf{R}$ and $s_1 = \mathbf{X} \cdot \mathbf{R}' + r$ to $Charlie$.
4. Bob sends $\mathbf{w}_2 = \mathbf{Y} + \mathbf{R}'$ and $s_2 = \mathbf{R} \cdot (\mathbf{Y} + \mathbf{R}) + r'$ to $Charlie$.
5. $Charlie$ computes $v = \mathbf{w}_1 \cdot \mathbf{w}_2 - s_1 - s_2$, and sends the result to $Alice$.
6. $Alice$ computes $\mathbf{X} \cdot \mathbf{Y} = v + (r + r')$.

This secure inner product uses additive masking to obfuscate each sensitive value before sharing with other parties.

We modify the above inner product protocol by first replacing the r and r' integers with vectors. Also rather than $Alice$ and Bob jointly generating $\mathbf{r}$, we propose that they

do this independently. That way, we can use the values in $\mathbf{r}$ as a one-time only random number only known to the party who generates it.

The modified algorithm is presented in Algorithm 1. All operations in Algorithm 1. are computed modulo a large prime q , and random numbers are chosen to be cryptographically secure. Due to simplicity of expression, the algorithm and other operations are not presented to show reduction modulo q , but it should be noted that it is implied.

Algorithm 1 Secure 3-Party Inner Product Protocol

procedure INITIALIZATION
  Set variable n, and publish to $Alice$ and Bob
  $Alice$ and Bob jointly generate random vectors $\mathbf{R}_A$ and $\mathbf{R}_B$, each of size n
  $Alice$ and Bob independently generate random vectors $\mathbf{r}_A$ and $\mathbf{r}_B$ respectively, each of size n
end procedure
procedure Alice
  for $i = 0 \rightarrow (n-1)$ **do**
    $\mathbf{W}_A[i] := \mathbf{X}[i] + \mathbf{R}_A[i]$
    $\mathbf{S}_A[i] := \mathbf{X}[i] \cdot \mathbf{R}_B[i] + \mathbf{r}_A[i]$
  end for
  $Alice$ sends $\mathbf{W}_A, \sum_{i=0}^{n-1} \mathbf{S}_A[i]$ to $Charlie$
end procedure
procedure Bob
  for $i = 0 \rightarrow (n-1)$ **do**
    $\mathbf{W}_B[i] := \mathbf{Y}[i] + \mathbf{R}_B[i]$
    $\mathbf{S}_B[i] := \mathbf{R}_A[i] \cdot (\mathbf{Y}[i] + \mathbf{R}_B[i]) + \mathbf{r}_B[i]$
  end for
  Bob sends $\mathbf{W}_B, \sum_{i=0}^{n-1} \mathbf{S}_B[i]$ to $Charlie$
  Bob sends $\sum_{i=0}^{n-1} \mathbf{r}_B[i]$ to $Alice$
end procedure
procedure Charlie
  $temp := \sum_{i=0}^{n-1} \mathbf{W}_A[i] \cdot \mathbf{W}_B[i]$
  $V := temp - \mathbf{S}_A[i] - \mathbf{S}_B[i]$
  Send V to $Alice$
end procedure
$Alice$ computes $\mathbf{X} \cdot \mathbf{Y} := V + \sum_{i=0}^{n-1} \mathbf{r}_A[i] + \sum_{i=0}^{n-1} \mathbf{r}_B[i]$

4.5. PREDICT

It is now clear that our goal is to compute Eq. 4.7 using secure multi-party computation. We adopt the modified version of the secure inner product protocol proposed by Du and Atallah [49], as presented in Algorithm 1. We replace three parties with the three stakeholders required for computing DST. The secure inner product protocol is used to compute $\sum_{i \in L(x)} a_i \cdot SNP_i^2$ and $\sum_{i \in L(x)} b_i \cdot SNP_i$ components and finally the aggregation of

the $\sum_{i \in L(x)} v_i$ component.

4.5.1. PRIVATE DST IN DTC MODEL

In our protocol as shown in Figure 4.2, there are 4 parties involved but only 3 parties required in computing the protocol. There is a non-collusion assumption on the parties, implying that no two parties are allowed to collaborate with others to learn more information than the protocol permits them.

The protocol by Ayday et al.[34] assumes that the genome data belongs to a patient, thereby presuming that the medical centre (genome analysing unit) is trusted to see the result of the DST. We adopt a contrary assumption. An individual ***P*** is any customer who seeks to learn information from their genome due to sheer curiosity. Our assumption makes it easier to appreciate why P might not necessarily want to share the end result of the susceptibility test with any party including the medical centre.

The protocol parties are as follows:

(i) The individual ***(P)***, is the customer whose genome is considered for analysis. P owns the SNPs required as input for the execution of the DST protocol, and will contribute them for computation in a privacy-preserving manner.

(ii) The Genome Analysing Unit ***(GA-Unit)***, represents a commercial platform that offers genome analyses as a service using a DTC model. This entity is considered to have a reputation that must be protected, therefore, should conform to ethical requirements within their field. To be simply put, ***GA-Unit*** is not assumed to be malicious. From Eq. 4.7, ***GA-Unit*** holds the values for $\{\mathbf{a}, \mathbf{b}, \mathbf{v}\}$ and would want to keep them private as well.

(iii) The Certified Sequencing Institution ***(CSI)***, handles sequencing of the genomes and transforming the biological sample of genomes to a digital format. The ***CSI*** is equally bounded to conform to ethical values. Although the ***CSI*** performs sequencing, it is not involved in the DST protocol and will not be further discussed during the course of computing DST.

(iv) The Processing Unit ***(PU)***, has a lot of processing resources that are required to handle huge computations. He is not to be trusted with unprotected genome data, but is assumed to follow the protocol and execute the expected computations. Since the ***PU*** does not contribute any input data to the evaluation of the function, it has no concern for privacy.

We further denote SNP_i as k_i, hence Eq. 4.7 is now re-written to:

$$S^X := \sum_{i \in L(x)} a_i k_i^2 - b_i k_i + v_i \tag{4.8}$$

It has been shown that Eq.4.8 can be computed using the secure inner product protocol. Let $\pi = |L(x)|$ be the number of SNPs needed for computing the susceptibility of a disease. Each SNP (represented by k_i) requires two multiplications and two additions.

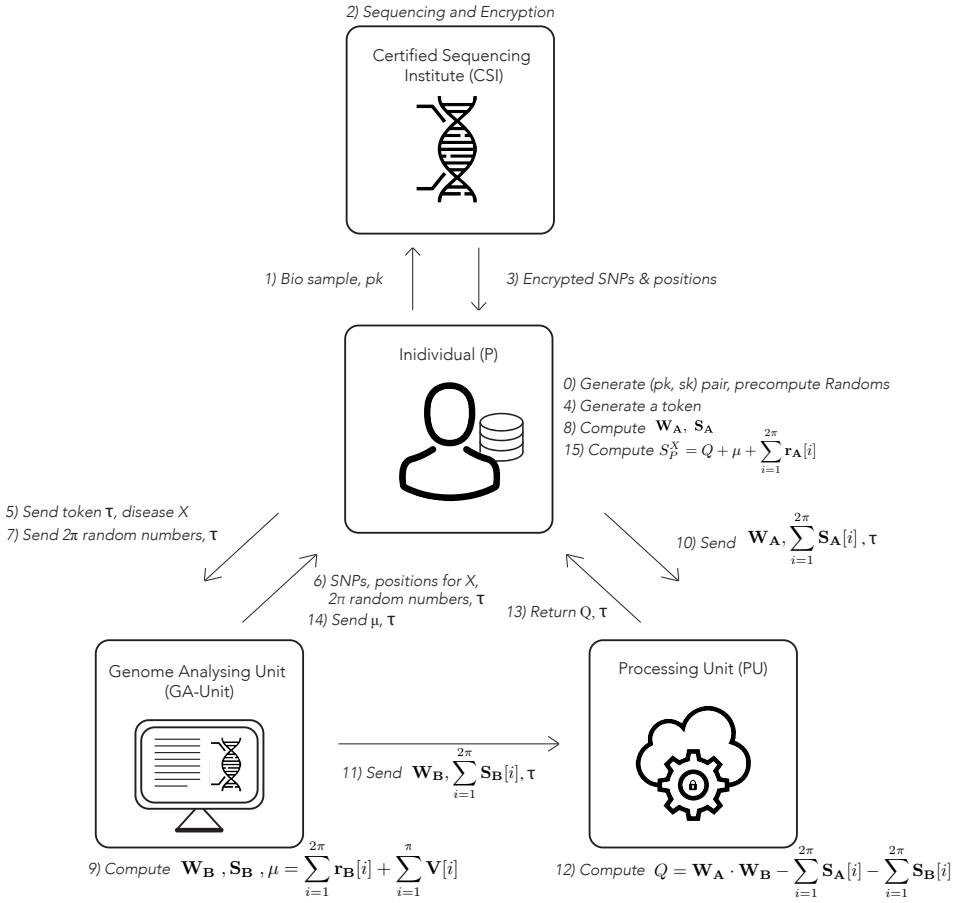

Figure 4.2: Private Disease Susceptibility Test on a DTC model

This means that if the number of SNPs required for computing a DST for a disease X is π . Therefore, the size of the vector contributed by both ***GA-Unit*** and ***P*** shall be of size 2π . The addition of v_i is taken care of by the aggregation property intrinsic in the secure inner product protocol.

4.5.2. PRIVACY AND SECURITY ASSUMPTIONS

Figure 4.2 shows the interactions and work-flow between parties within our proposed protocol. Our protocol is designed with the following security assumptions in mind:

a) The individual (***P***) is aware of the sensitivity of his genome data and needs to utilise the computational ability of the processing unit or cloud infrastructure and the knowledge of the genome analysing unit, without leaking any sensitive information to the ***PU*** and ***GA-Unit***. On the contrary, ***P*** does not pose any privacy threat

to ***PU*** during the course of execution of the protocol.

b) ***GA-Unit*** is in possession of the individual weights of SNPs for diseases. The weights are considered trade secrets and should equally be protected from ***P*** and ***PU***, while being used to privately compute the susceptibility of ***P*** to a disease.

c) The device on which ***P***'s data are stored, is considered secure and only accessible with ***P***'s permission.

4.5.3. PROTOCOL DESCRIPTION

A detailed description of steps in the proposed protocol is as follows:

- **Step 0:** ***P*** generates a pair of cryptographic keys, consisting of a private key and a public key. The public key is made public to ***CSI*** only, for encrypting the digital sequence of ***P***'s genome. This case is slightly different if we adopt the symmetric key option, where ***P*** generates a single private key with the hope of sharing it with ***CSI***.

- **Step 1:** ***P*** sends a biological sample of his genome to the ***CSI*** for sequencing, alongside his public key (or private key in the case of a symmetric crypto scheme).

- **Step 2:** ***CSI*** receives the sample and sequences the genome, produces it *in silico* and encrypts the sequence with the public key of ***P***.

- **Step 3:** ***CSI*** returns the encrypted SNPs and the corresponding locations to ***P***. The ***CSI*** is also obligated to securely delete all copies and traces of the genome data. This is necessary because the ***CSI*** is not expected to further participate actively in the remaining part of the protocol.

- **Step 4:** At this point, ***P*** is in full possession and control of his encrypted SNPs, and our assumption allows that only the individual has any possible copy of his SNPs. Now, ***P*** performs a one-time only decryption of the encrypted data, and saves the SNP related data in his secure device. We assume that such a device is protected and is private to only ***P***. For a susceptibility test to be initiated by ***P***, he is first required to generate a random token τ (a 160-bit value). This token is unique and used to reference every instance of a disease susceptibility test.

- **Step 5:** It is important to note that our assumption presumes that ***P*** is online and is therefore able to carry out his part of the protocol. It is expected that ***P*** is active and is able to locate a publicly available unique disease identifier (ID) published by ***GA-Unit***. The public identifiers (IDs) may be published on a website, and ***P*** can then send a token τ and the disease ID to ***GA-Unit*** after necessary authentication.

- **Step 6:** The ***GA-Unit*** will verify the request from ***P***, and then responds with the SNP positions for the disease X, where X is the corresponding disease for the supplied ID. The ***GA-Unit*** will always demand a constant size π of SNPs. This is to make it more difficult for an observer to infer what disease ***P*** is interested in. The extra (dummy) SNPs are pertinent to obfuscate the actual SNPs relevant to the disease. The set of dummy SNPs are to be deterministic for any disease. As to prevent

P from distinguishing real SNPs from dummy SNPs. Also, the ***GA-Unit*** sends 2π random numbers to ***P***, called vector $\mathbf{R}_B$. These randoms will be used for the secure inner product protocol. Finally, ***GA-Unit*** generates vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{v}$.

- **Step 7:** When ***P*** receives the SNP positions from ***GA-Unit***, he generates 2π random numbers and sends the vector to ***GA-Unit***. We denote this random number vector as $\mathbf{R}_A$.
- **Step 8:** In order for both ***P*** and ***GA-Unit*** to compute Eq. 4.8, they agree to split the equation into two parts $a_i k_i^2$ and $(-b_i k_i + v_i)$. ***P*** generates another set of 2π random numbers, denoted as $\mathbf{r}_A$. Generates the vector $\mathbf{W}_A$ by computing $\mathbf{W}_A[i] := k_i^2 + \mathbf{R}_A[i]$ and $\mathbf{W}_A[i+\pi] := -k_i + \mathbf{R}_A[i+\pi]$. The vector $\mathbf{S}_A$ is also generated by computing $\mathbf{S}_A[i] := k_i^2 \cdot \mathbf{R}_B[i] + \mathbf{r}_A[i]$ and $\mathbf{S}_A[i+\pi] := -k_i \cdot \mathbf{R}_B[i+\pi] + \mathbf{r}_A[i+\pi]$.
- **Step 9:** The ***GA-Unit*** first has to generate a vector of 2π random numbers, which we refer to as $\mathbf{r}_B$. Then, just like ***P*** generated $\mathbf{W}_A, \mathbf{S}_A$, ***GA-Unit*** generates $\mathbf{W}_B, \mathbf{S}_B$ as follows: $\mathbf{W}_B[i] := a_i + \mathbf{R}_B[i]$ and $\mathbf{W}_B[i+\pi] := b_i + \mathbf{R}_B[i]$. Then the vector $\mathbf{S}_B$ is generated as $\mathbf{S}_B[i] := \mathbf{R}_A[i](a_i + \mathbf{R}_B[i]) + \mathbf{r}_B[i]$ and $\mathbf{S}_B[i+\pi] := \mathbf{R}_A[i+\pi](b + \mathbf{R}_B[i+\pi]) + \mathbf{r}_B[i+\pi]$. Lastly, ***GA-Unit*** computes the variable $\mu := \sum_{i=1}^{2\pi} \mathbf{r}_B[i] + \sum_{i=1}^{\pi} \mathbf{v}[i]$.
- **Step 10:** ***P*** transmits the values $\mathbf{W}_A$, $\sum_{i=1}^{2\pi} \mathbf{S}_A[i]$ to ***PU***.
- **Step 11:** The ***GA-Unit*** transmits the values $\mathbf{W}_B$, $\sum_{i=1}^{2\pi} \mathbf{S}_B[i]$ to ***PU***.
- **Step 12:** ***PU*** computes $Q := \mathbf{W}_A \cdot \mathbf{W}_B - \sum_{i=1}^{2\pi} \mathbf{S}_A[i] - \sum_{i=1}^{2\pi} \mathbf{S}_B[i]$.
- **Step 13:** ***PU*** sends Q to ***P***.
- **Step 14:** The ***GA-Unit*** sends μ to ***P***.
- **Step 15:** Finally, ***P*** computes $S_P^X := Q + \mu + \sum_{i=1}^{2\pi} \mathbf{r}_A[i]$.

4.5.4. CORRECTNESS

The proof of correctness for the computation of S^X in PREDICT inherits the proof of correctness of the secure inner product protocol. The original protocol by Du and Atallah [33] computes an inner product of vectors between two mistrusting parties. However, our equation is of the form $\sum_{i=1}^{\pi} a_i \cdot k_i^2 - b_i \cdot k_i + v_i$, hence we require a slight modification to the original protocol. Variables $(-\mathbf{k}, \mathbf{k^2})$ are contributed by ***P*** while variables $(\mathbf{a}, \mathbf{b}, \mathbf{v})$ belong to the ***GA-Unit***. Each party then holds a vector for their variables, and each vector is of size π . First, we split the equation into two parts that can each be executed using an instance of the secure inner product protocol. One half of the equation is $\sum_{i=1}^{\pi} a_i \cdot k_i^2$ and the other $\sum_{i=1}^{\pi} (-k_i \cdot b_i + v_i)$. Nevertheless, since we are computing a modified version

of the form $\sum_{i=1}^{\pi}(b_i \cdot k_i + v_i)$, it suffices to show that a circuit that correctly computes $\sum_{i=1}^{\pi} a_i \cdot k_i$, can be modified to compute $\sum_{i=1}^{\pi}(a_i \cdot k_i + v_i)$ without loss of security.

Since **v** is considered to be part of ***GA-Unit***'s trade secret [33], we have to transfer **v** to ***P*** without revealing the value *in clear*. We do this by additively masking **v** values with random numbers. Specifically, we use the random numbers $\mathbf{r}_B$ which is known to ***GA-Unit*** but oblivious to ***P*** in **Step 9** to mask the values of **v**. This operation still preserves the correctness of the computation. Having established that $\sum_{i=1}^{\pi}(-k_i \cdot b_i + v_i)$ can be computed using the secure inner product protocol, we perform one more step. We merge the vectors of $\sum_{i=1}^{\pi} a_i \cdot k_i^2$ and $\sum_{i=1}^{\pi}(-k_i \cdot b_i + v_i)$ by appending the latter to the former. We produce a new vector of size 2π, with which we compute the secure inner product. Thus, an equation of the form $\sum_{i=1}^{\pi} a_i \cdot k_i^2 - b_i \cdot k_i + v_i$, can be correctly computed using PREDICT.

Table 4.1: DATA COMMUNICATION COMPLEXITY FOR THE PROPOSED PROTOCOL

		Received (bits)			
Sent (bits)		***CSI***	***P***	***PU***	***GA-Unit***
	CSI	–	3) AES: 51MB, RSA: 47MB	–	–
	P	1) AES: 128, RSA: 2065	–	10) $266\pi + 297 + \log_2(2\pi)$	5) 288
	PU		13) $426 + \log_2(2\pi)$	–	7) $264\pi + 160$
	GA-Unit	–	6) $298\pi + 160$ 14) $293 + \log_2(2\pi)$	11) $266\pi + 426 + \log_2(2\pi)$	–

4.6. COMPLEXITY ANALYSIS

In Table 4.1, we present an overview of the communication complexity of our protocol. All units of data transfer are in bits, except for Step 3 of Figure 4.2, where the data is represented in megabyte (MB). Let $\mathcal{N}$ denote the plaintext size (in bits) for the crypto scheme adopted, which provides an appropriate security level (default value: κ = 112-bits). An example of a SNP reference is *rs138055828*, we only consider the number numeric part of the reference code. All random numbers generated are of size 132-bits, while each of the variables $(\mathbf{a}, \mathbf{b}, \mathbf{v})$ contributed by ***GA-Unit*** is of size 20 bits. Let M represent the number of SNPs that can be packed into a single ciphertext (without any SNP overflowing into a new block of ciphertext). Consequently,

$$M_{RSA} := \left\lfloor \frac{\mathcal{N} - 128}{36} \right\rfloor = \left\lfloor \frac{2048 - 128}{36} \right\rfloor = 53 \text{ , and}$$
$$M_{AES} := \left\lfloor \frac{\mathcal{N}}{36} \right\rfloor = \left\lfloor \frac{128}{36} \right\rfloor = 3 \text{ ,} \tag{4.9}$$

where 128-bits are required for RSA padding.

If an individual has 10 million SNPs as reported by The National Library of Medicine[17], then let t denote the number of packed ciphertext blocks produced on encrypting 10

million records (SNPs).

$$t_{RSA} := \left\lceil \frac{10,000,000}{M_{RSA}} \right\rceil = \left\lceil \frac{10,000,000}{53} \right\rceil = 188,680\,, \text{ and}$$
$$t_{AES} := \left\lceil \frac{10,000,000}{M_{AES}} \right\rceil = \left\lceil \frac{10,000,000}{3} \right\rceil = 3,333,334\,. \tag{4.10}$$

For any individual who has 10 million SNPs, we require 188680 units of RSA ciphertexts, where a single ciphertext is of size 2048 bits. From Table 4.1, it is clear that the communication complexity is linear in the number of SNPs required for a DST. In fact, even for 10 million SNPs, the protocol will require less than 1GB of data transfer during the DST computation.

Table 4.2: COMPUTATION COMPLEXITY

	P	***GA-Unit***	***PU***
Addition	$6\pi+2$	12π	2
Multiplication	2π	2π	2π

The computational overhead of our protocol is shown in Table 4.2. Although the number of operations are provided for simplicity, we note that not all operations are of the same complexity. For instance, addition counts in Table 4.2 include adding a 2-bit and a 133-bit numbers, as well as adding a 132-bit and a 265-bit number. From Table 4.2, it can be deduced that the computational complexity for computing a disease susceptibility test is linear in the size (π) of the SNPs relevant for computing such a test.

4.7. OPTIMIZATION OF PREDICT.

Firstly, a variant setting of this protocol could be achieved by altering the flow of operations halfway into the protocol. For instance, rather than have the ***GA-Unit*** send $\mu := \sum_{i=1}^{2\pi} \mathbf{r}_B[i] + \sum_{i=1}^{\pi} \mathbf{v}[i]$ to ***P***, ***GA-Unit*** can send $\mu := \sum_{i=1}^{2\pi} \mathbf{r}_B[i]$ to ***P*** and send the other value to ***PU*** indirectly by modifying **Step 11:** to $\sum_{i=1}^{2\pi} \mathbf{S}_B[i] - \sum_{i=1}^{2\pi} \mathbf{r}_B[i] - \sum_{i=1}^{\pi} \mathbf{v}[i]$. This will reduce the computation and communication overhead on ***P*** and place it on ***PU*** who is assumed to have sufficient resources. This will save both computation and communication costs for ***P***. Moreover, the correctness of the protocol will still hold. Adopting this optimization will offer significant improvement where large number of SNPs are required and the security level is equally very high. However, for the default setting of this protocol, such an optimization will not offer a significant improvement.

Secondly, due to the time it takes to generate random numbers, ***P*** and ***GA-Unit*** might have to pre-generate random numbers as part of the preprocessing phase. This saves time and allow for the rest of the protocol to be executed seamlessly, with only basic operations.

Thirdly, another optimization step is to adopt a symmetric key crypto scheme for encryption in **Step 1**. By this, we achieve a reduction in bits of the ciphertext being transferred from the ***CSI*** to ***P*** in **Step 3**. Since a symmetric crypto scheme will offer faster

encryption and decryption operations, this offers a computational reduction in the time it takes ***P*** to decrypt his sequence. Recall that the decryption operation has to be performed only once. Thereafter, the data will be stored *in clear* within the secure device of ***P***. However, this approach requires that the ***CSI*** and ***P*** will share P's secret key.

Finally, we recommend a data packing technique for encrypting the SNPs. Every SNP can be represented with 36 bits, given that 2 bits represent the SNP value (0, 1, 2) refer to Section 4.4.3. The other 34 bits are for referencing the SNP, otherwise known as the SNP position. Data packing will group more than one SNP into a single block of ciphertext, thereby optimizing the time required to decrypt and access the entire SNPs of an individual.

4.8. DISCUSSION

PREDICT differs from the proposal by Ayday et al.[34] as follows:

- Our primary aim is to protect the privacy of an individual's genome data from all other entities in the protocol, while being able to harness their abilities to test for disease susceptibility. Only the individual ***P*** is allowed to view the result of every susceptibility test. However, Ayday et al's protocol does not seek to protect the privacy of the individual's genome data from the genome analyzing unit, which results from their assumption that the individual is a patient and the genome analyzing unit is a doctor in a medical institution.

- We propose that genome data should be stored in a dedicated piece of hardware, that should only be accessible by the individual ***P***. This allows ***P*** to have full control of his digital genome data and also provides him the freedom to change the cryptographic keys and other security measure when necessary. These can be done without incurring much cost or informing a third party about the intentions to make changes to the cryptographic keys. Our choice to decentralize the genome data storage helps to reduce the risk of targeting a central cloud storage infrastructure.

- Our protocol guarantees ***P***'s independence from a medical unit. Thereby, realizing our aim of providing privacy for curiosity driven individuals, and at the same time offering a DTC service for disease susceptibility testing using genetic data. The protocol by Ayday et al. is not designed to target a DTC scenario.

- In our setting, the obfuscation of the SNP positions and values are meant to be computed by ***P*** and sent to the ***PU***. Replacing encryption with randomization eliminates the expensive homomorphic operations for all parties. The ***GA-Unit*** is not expected to possess the processing power required to compute over encrypted data. However, introducing randomisation as opposed to encryption requires that we have a secure means for generating fresh and cryptographically secure random numbers. The hardware on which ***P*** stores his genome data is assumed to provide such requirements. Random number can be pre-generated and securely stored on such devices.

- Our protocol does not leak SNPs of ***P*** to the processing unit. Since the processing unit cannot distinguish the real SNPs from the dummy SNPs.

- Our protocol offers reduced storage cost to the individual. This is a result of storing encrypted data using data packing techniques.

4.9. IMPLEMENTATION

Here, we present the implementation of PREDICT as a prototype using basic tools. Our implementation uses simulated data rather than real dataset, since a real dataset can always be substituted whenever such data is available. We simulate ten thousand SNPs values as random numbers uniformly distributed between 0 and 2, to represent input data for the customer. The weights are equally simulated and scaled to integer values, which represent the input data of the commercial platform. The prototype of PREDICT was implemented in C++, using NTL and GMP as dependency libraries. All codes are written and executed as sequentially. Our implementation was tested on a computer with Intel Core i7-4770, 3.40 GHz, 16 GB of RAM, and 64-bit version of Ubuntu 18.04 LTS. The prototype implementation shows that PREDICT scales linearly in the size of SNP values required for a DST. As an example, computing DST using 10,000 SNP only takes about 96 *milliseconds*. In Table 4.3 we show comparison of our protocol with that by Ayday et al.[34].

Table 4.3: COMPUTATION COMPLEXITY

	Ayday et al.	**PREDICT**
Technique	Additive HE	Masking
SNP Storage	Centralized	Decentralized
Privacy Leaks	Yes	No
Performance	2 mins/10 SNPs	96 ms/10,000 SNPs

4.10. PRIVACY ANALYSES

Our Direct-to-Consumer DST protocol is described in the semi-honest (honest but curious) security model. Also, there is a non-collusion assumption on the entities (***P***, ***GA-Unit***, ***PU***, ***CSI***) apart from those explicitly specified within the protocol. Actually, the value π is chosen as follows: Let $\mathbb{D} = \{X_1, \ldots, X_m\}$ be the set of all diseases, and $|X_i|$ represents the number of SNPs that are associated with a disease X_i . If $\top := max\{|X_i|\ ,\ \forall\ X_i \in \mathbb{D}\}$, then $\pi := \top + \kappa$.

The privacy of data is argued using simulation-based security reduction [46, 50, 51].

Negligible function: A function $\mu(\cdot)$ is negligible if for every positive polynomial $p(\cdot)$ and all sufficiently large $\kappa \in \mathbb{N}$, it holds that $\mu(\kappa) < 1/p(\kappa)$.

Computational indistinguishability: Given that $a \in \{0,1\}^*$ and κ is security parameter, let $X = X(a,\kappa)$ and $Y = Y(a,\kappa)$ be two probability ensembles. X and Y are said to be computationally indistinguishable, denoted by $X \overset{c}{\equiv} Y$, if for every non-uniform probabilistic polynomial-time (PPT) algorithm D , there exists a negligible function $\mu(\cdot)$ such

that

$$|\Pr[D(X(a,\kappa))=1]-\Pr[D(Y(a,\kappa))=1]|\leq\mu(\kappa)\,. \tag{4.11}$$

$\overset{s}{\equiv}$ denotes statistical indistinguishability. **Security**: Let $f=(f_1,f_2)$ be an ideal functionality and let Π be a real-world two-party protocol for computing f. Where f_1,f_2 denote the results corresponding to parties 1 and 2 respectively on running f. The view of the party $i\in\{1,2\}$ during the execution of Π on input (a,b) and security parameter κ is denoted by $\mathbf{view}_i^{\Pi}(a,b,\kappa):=(w,r^i;m_1^i,\ldots,m_t^i)$, where $w\in(a,b)$, and r^i is the content of party i's internal random tape, and m_j^i represents the j-th message received.

The output of party i during the execution of Π on the inputs (a,b) with security parameter κ is denoted by, $\mathbf{output}_i^{\Pi}(a,b,\kappa)$ and can be computed from its own view of the execution. The joint output of both parties is denoted by

$$\mathbf{output}^{\Pi}(a,b,\kappa)=(\mathbf{output}_1^{\Pi}(a,b,\kappa),\mathbf{output}_2^{\Pi}(a,b,\kappa))\,.$$

We say that Π securely computes f in the presence of semi-honest adversaries if there exists PPT algorithms $\mathscr{S}_1$ and $\mathscr{S}_2$ such that:

$$\{\mathscr{S}_1(1^\kappa,a,f_1(a,b)),f(a,b)\}\overset{c}{\equiv}\{(\mathbf{view}_1^{\Pi}(a,b,\kappa),\mathbf{output}^{\Pi}(a,b,\kappa))\}\,.$$

$$\{\mathscr{S}_2(1^\kappa,b,f_2(a,b)),f(a,b)\}\overset{c}{\equiv}\{(\mathbf{view}_2^{\Pi}(a,b,\kappa),\mathbf{output}^{\Pi}(a,b,\kappa))\}\,. \tag{4.12}$$

Although we have provided definitions for a two-party computation, the remainder of the security proof is extended for a three-party computation without loss of generality. The ideal functionality f takes ordered inputs from ***P***, ***GA-Unit*** and ***PU*** respectively. The aim of the proof is to show that the view of a PPT adversary $\mathscr{A}$ in the real-world execution of the protocol Π, is computationally indistinguishable from the view of a simulator $\mathscr{S}_i$ for $i\in\{1,2,3\}\equiv\{\boldsymbol{P},\boldsymbol{GA\text{-}Unit},\boldsymbol{PU}\}$ in the ideal world execution of the protocol f. Specifically, we consider three distinct scenarios where an adversary compromises each of the parties in order to gain information about the private data of other parties.

Scenario 1: Let us assume that ***P*** has been compromised by an adversary $\mathscr{A}$. Then, $\mathscr{S}_1$ is provided with the inputs and outputs of ***P***, and is required to simulate the view: Note that the privacy assets are the weights of SNPs to diseases, which are trade secrets and denoted as the vectors $(\mathbf{a},\mathbf{b},\mathbf{v})$. Refer to Figure 4.2 and Eq.4.8 for details on variables. Since ***PU*** does not contribute any input, we do not have to worry about ***PU***'s privacy. Let $\perp$ denote an empty string. The DST protocol Π securely and privately computes the DST functionality $f((\mathbf{k},\mathbf{k^2}),(\mathbf{a},\mathbf{b},\mathbf{v}),\perp)=(S_P^X,\perp,\perp)$ in the presence of any honest-but-curious PPT adversary.

$$\mathbf{view}_1^{\Pi}(((\mathbf{k},\mathbf{k}^2),(\mathbf{a},\mathbf{b},\mathbf{v})),\kappa):=(1^\kappa,r^P,\mathbf{k},\mathbf{k}^2,\mathbf{R}_B,Q,\mu)\,, \tag{4.13}$$

where r^P is a uniformly distributed random tape.

In order to simulate the view of ***P***, $\mathscr{S}_1$ does the following:

1. $\mathscr{S}_1$ starts the protocol with his inputs $(\mathbf{k}, \mathbf{k}^2)$, and generates the vectors of random numbers $\mathbf{r}'_A, \mathbf{R}'_A$. Observe that the randoms are different from those generated by an honest ***P***.

2. For Step 6: $\mathscr{S}_1$ generates the vector of randoms $\mathbf{R}'_B$, to simulate incoming input from ***GA-Unit***.

3. For Step 13: In order to simulate Q as received from ***PU***, $\mathscr{S}_1$ first generates vectors $\mathbf{W}'_B$, $\mathbf{S}'_B$ such that the elements of the vector $\mathbf{W}'_B$ and $\mathbf{S}'_B$ come from the same space as elements of $\mathbf{W}_B$ and $\mathbf{S}_B$ respectively. Then, compute $\mathbf{W}'_A$ and $\mathbf{S}'_A$ as prescribed in Step 8. Finally, compute $Q' = \mathbf{W}'_A \cdot \mathbf{W}'_B - \sum_{i=1}^{2\pi} \mathbf{S}'_A - \sum_{i=1}^{2\pi} \mathbf{S}'_B$.

4. For Step 14: $\mathscr{S}_1$ computes $\mu' = \sum_{i=1}^{2\pi} \mathbf{r}'_B - \sum_{i=1}^{\pi} \mathbf{v}'$. The vector $\mathbf{v}'$ is generate from the same space as $\mathbf{v}$.

From the above, the simulated view of $\mathscr{S}_1$ can be expressed as:

$$\mathscr{S}_1(1^\kappa, \mathbf{k}, \mathbf{k}^2, f_1((\mathbf{k}, \mathbf{k}^2), (\mathbf{a}, \mathbf{b}, \mathbf{v}))) := (1^\kappa, r^P, \mathbf{k}, \mathbf{k}^2, \mathbf{R}'_B, Q', \mu') \cdot \tag{4.14}$$

From Equations 4.13 & 4.14, we conclude that

$$\mathscr{S}_1(1^\kappa, \mathbf{k}, \mathbf{k}^2, f_1((\mathbf{k}, \mathbf{k}^2), (\mathbf{a}, \mathbf{b}, \mathbf{v}))) \stackrel{s}{\equiv} \mathbf{view}_1^{\Pi}(((\mathbf{k}, \mathbf{k}^2), (\mathbf{a}, \mathbf{b}, \mathbf{v})), \kappa) \cdot$$

For any PPT distinguisher D ,

$$\begin{aligned} &Pr[D(1^\kappa, r^{S_1}, \mathbf{k}, \mathbf{k}^2, \mathbf{R}'_B, Q', \mu') = 1] - \\ &Pr[D(1^\kappa, r^P, \mathbf{k}, \mathbf{k}^2, \mathbf{R}_B, Q, \mu) = 1] \leq \frac{1}{\mu(\kappa)} \cdot \end{aligned} \tag{4.15}$$

Scenario 2: We assume that ***GA-Unit*** is compromised and the aim is to learn the values of the SNPs which are the vectors $\mathbf{k}, \mathbf{k}^2$.

$$\mathbf{view}_2^{\Pi}(((\mathbf{k}, \mathbf{k}^2), (\mathbf{a}, \mathbf{b}, \mathbf{v})), \kappa) := (1^\kappa, r^G, \mathbf{a}, \mathbf{b}, \mathbf{v}, \mathbf{R}_A) \cdot \tag{4.16}$$

where r^G is a uniformly distributed random tape. In order for $\mathscr{S}_1$ to simulate the operations of ***GA-Unit***, the following steps occur:

1. $\mathscr{S}_2$ is provided with the inputs of ***GA-Unit*** and the disease X. These are the vectors ($\mathbf{a}$, $\mathbf{b}$, $\mathbf{v}$).

2. For Step 7: $\mathscr{S}_2$ generates a vector of random numbers $\mathbf{R}'_A$, which should be sampled from the same space as $\mathbf{R}_A$.

No other input is received by ***GA-Unit***, and this makes the proof trivial. The simulated view of $\mathscr{S}_2$ is then expressed as:

$$\mathscr{S}_2(1^\kappa, \mathbf{a}, \mathbf{b}, \mathbf{v}, f_2((\mathbf{k}, \mathbf{k}^2), (\mathbf{a}, \mathbf{b}, \mathbf{v}))) := (1^\kappa, r^{S_2}, \mathbf{a}, \mathbf{b}, \mathbf{v}, \mathbf{R}'_A) \cdot \tag{4.17}$$

From Equations 4.16 & 4.17, we have that

$$\mathscr{S}_2(1^\kappa, \mathbf{a}, \mathbf{b}, \mathbf{v}, f_2((\mathbf{k}, \mathbf{k}^2), (\mathbf{a}, \mathbf{b}, \mathbf{v}))) \stackrel{s}{\equiv} \mathbf{view}_2^{\Pi}(((\mathbf{k}, \mathbf{k}^2), (\mathbf{a}, \mathbf{b}, \mathbf{v})), \kappa).$$

For any PPT distinguisher D,

$$Pr[D(1^\kappa, r^{S_2}, \mathbf{a}, \mathbf{b}, \mathbf{v}, \mathbf{R}'_A) = 1] - Pr[D(1^\kappa, r^G, \mathbf{a}, \mathbf{b}, \mathbf{v}, \mathbf{R}_A) = 1] \leq \frac{1}{\mu(\kappa)}.$$

Scenario 3: We assume that ***PU*** is compromised by an adversary $\mathscr{A}$. The aim is to learn the private values of ***P*** and ***GA-Unit*** which include $(\mathbf{k}, \mathbf{k}^2)$, (**a, b, c**) .

$$\mathbf{view}_3^{\Pi}(((\mathbf{k}, \mathbf{k}^2), (\mathbf{a}, \mathbf{b}, \mathbf{v})), \kappa) := (1^\kappa, r^{PU}, \perp, \mathbf{W}_A, \mathbf{W}_B, \sum_{i=1}^{2\pi} \mathbf{S}_A, \sum_{i=1}^{2\pi} \mathbf{S}_B). \tag{4.18}$$

where r^{PU} is a uniformly distributed random tape. For simulator $\mathscr{S}_3$ to simulate the view of ***PU***, the following steps are followed:

1. $\mathscr{S}_3$ is provided with the security parameter κ.
2. For Step 10: $\mathscr{S}_3$ generates the vector $\mathbf{W}'_A$ from the same space as $\mathbf{W}_A$. Then, he generates a vector $\mathbf{S}'_A$ and computes the value $\sum_{i=1}^{2\pi} \mathbf{S}'_A$.
3. For Step 11: $\mathscr{S}_3$ generates the vector $\mathbf{W}'_B$ from the same space as $\mathbf{W}_B$. Then, he generates a vector $\mathbf{S}'_B$ and computes the value $\sum_{i=1}^{2\pi} \mathbf{S}'_B$.

The simulated view of $\mathscr{S}_3$ is then expressed as:

$$\mathscr{S}_3(1^\kappa, \perp, f_3((\mathbf{k}, \mathbf{k}^2), (\mathbf{a}, \mathbf{b}, \mathbf{v}))) := (1^\kappa, r^{S_3}, \perp, \mathbf{W}'_A, \mathbf{W}'_B, \sum_{i=1}^{2\pi} \mathbf{S}'_A, \sum_{i=1}^{2\pi} \mathbf{S}'_B). \tag{4.19}$$

From Equations 4.18 & 4.19, we have that

$$\mathscr{S}_3(1^\kappa, \perp, f_3((\mathbf{k}, \mathbf{k}^2), (\mathbf{a}, \mathbf{b}, \mathbf{v}))) \stackrel{s}{\equiv} \mathbf{view}_3^{\Pi}(((\mathbf{k}, \mathbf{k}^2), (\mathbf{a}, \mathbf{b}, \mathbf{v})), \kappa).$$

For any PPT distinguisher D,

$$Pr[D(1^\kappa, r^{S_3}, \perp, \mathbf{W}'_A, \mathbf{W}'_B, \sum_{i=1}^{2\pi} \mathbf{S}'_A, \sum_{i=1}^{2\pi} \mathbf{S}'_B) = 1] - Pr[D(1^\kappa, r^{PU}, \perp, \mathbf{W}_A, \mathbf{W}_B, \sum_{i=1}^{2\pi} \mathbf{S}_A, \sum_{i=1}^{2\pi} \mathbf{S}_B) = 1] \leq \frac{1}{\mu(\kappa)}. \tag{4.20}$$

4.11. CONCLUSION

This paper presents a protocol that blends the direct-to-consumer genetic testing model and the need to protect consumers' privacy. We have shown that a cryptographic solution to the problem is possible and implementable for practical use. Under our proposed protocol, the use of one-time-only masking is deployed to obfuscate sensitive data. We show that our proposed protocol provides security and privacy for both the genome data owners and the commercial platform while they collaborate to perform a disease susceptibility test. The design we propose introduces less work for all parties, as they are required to compute over randomized data instead of encrypted data. Our approach eliminates the storage of encrypted data on a third-party cloud infrastructure as was suggested by some earlier works. Rather, we recommend decentralizing the storage of the genome data and only allowing for storage on a device owned and controlled by genome owners. Distributing the data also eliminates a single point of failure. Our proposal allows any customer to easily update newly discovered SNPs. A prototype implementation shows that with as much as 10,000 SNPs, the DST can be computed in about 96 $milliseconds$ on a commodity hardware, ignoring the network transfer time. This outperforms other existing homomorphic encryption based approaches where computational complexity is dominated by homomorphic operations. Our proposal scales linearly in the size of the SNPs, and has shown to be practicable in the wild. Finally, we mention that our solution does not plug the analog hole. For instance, it does not protect a scenario where an attacker is able to physically force an individual to reveal his SNP values. Such an attack can be compared to an adversary retrieving a biological sample from the individual, to sequence the genome.

REFERENCES

[1] A. Sboner, X. J. Mu, D. Greenbaum, R. K. Auerbach, and M. B. Gerstein, *The real cost of sequencing: higher than you think!* Genome biology **12**, 125 (2011).

[2] J. Shendure, S. Balasubramanian, G. M. Church, W. Gilbert, J. Rogers, J. A. Schloss, and R. H. Waterston, *Dna sequencing at 40: past, present and future,* Nature **550**, 345 (2017).

[3] J. E. Wylie and G. P. Mineau, *Biomedical databases: protecting privacy and promoting research,* Trends in biotechnology **21**, 113 (2003).

[4] E. Ayday, M. Humbert, J. Fellay, M. Laren, P. Jack, J. Rougemont, J. L. Raisaro, A. Telenti, and J.-P. Hubaux, *Protecting personal genome privacy: Solutions from information security*, Tech. Rep. (EPFL, 2012).

[5] D. L. Selwood, *Beyond the hundred dollar genome–drug discovery futures,* Chemical biology & drug design **81**, 1 (2013).

[6] D. McMorrow, *The $100 genome: Implications for the DoD*, Tech. Rep. (DTIC Document, 2010).

[7] L. A. Hindorff, P. Sethupathy, H. A. Junkins, E. M. Ramos, J. P. Mehta, F. S. Collins, and T. A. Manolio, *Potential etiologic and functional implications of genome-wide*

association loci for human diseases and traits, Proceedings of the National Academy of Sciences **106**, 9362 (2009).

[8] M. K. Leung, A. Delong, B. Alipanahi, and B. J. Frey, *Machine learning in genomic medicine: a review of computational problems and data sets,* Proceedings of the IEEE **104**, 176 (2016).

[9] K. A. Wetterstrand, *Dna sequencing costs: data from the nhgri genome sequencing program (gsp). 2013,* URL http://www. genome. gov/sequencingcosts (2016).

[10] M. S. Reuter, S. Walker, B. Thiruvahindrapuram, J. Whitney, I. Cohn, N. Sondheimer, R. K. Yuen, B. Trost, T. A. Paton, S. L. Pereira, *et al.*, *The personal genome project canada: findings from whole genome sequences of the inaugural 56 participants,* Canadian Medical Association Journal **190**, E126 (2018).

[11] G. Church, *The race for the $1000 genome,* Science **311** (2006).

[12] Helix, *Helix,* (2018), `https://www.helix.com` Online; accessed January, 2018.

[13] MyHeritage, *Myheritage,* (2018), `https://www.myheritage.nl` Online; accessed November, 2018.

[14] 23andMe, *23andme,* (2018), `https://www.23andme.com/en-eu/` Online; accessed January, 2018.

[15] M. Fox, *Drug giant glaxo teams up with dna testing company 23andme,* (July, 2018), `https://www.nbcnews.com/health/health-news/drug-giant-glaxo-teams-dna-testing-company-23andme-n894531` Online; accessed November, 2018.

[16] S. Prashad and S. Srikanthan, *23andme: Building a genetically-sound company,* (April, 2018), `https://iveybusinessreview.ca/6346/23andme-building-genetically-sound-company/` Online; accessed November, 2018.

[17] NLM, *Single nucleotide polymorphism,* (2018), `https://ghr.nlm.nih.gov/primer/genomicresearch/snp` Online; accessed January, 2018.

[18] K. Michailidou, S. Lindström, J. Dennis, J. Beesley, S. Hui, S. Kar, A. Lemaçon, P. Soucy, D. Glubb, A. Rostamianfar, *et al.*, *Association analysis identifies 65 new breast cancer risk loci,* Nature **551**, 92 (2017).

[19] M. Watanabe, *Polymorphic cyp genes and disease predisposition?what have the studies shown so far?* Toxicology letters **102**, 167 (1998).

[20] S. Kathiresan, O. Melander, D. Anevski, C. Guiducci, N. P. Burtt, C. Roos, J. N. Hirschhorn, G. Berglund, B. Hedblad, L. Groop, *et al.*, *Polymorphisms associated with cholesterol and risk of cardiovascular events,* New England Journal of Medicine **358**, 1240 (2008).

[21] A. Thomas, N. J. Camp, J. M. Farnham, K. Allen-Brady, and L. A. Cannon-Albright, *Shared genomic segment analysis. mapping disease predisposition genes in extended pedigrees using snp genotype assays,* Annals of human genetics **72**, 279 (2008).

[22] X. Wan, C. Yang, Q. Yang, H. Xue, N. L. Tang, and W. Yu, *Megasnphunter: a learning approach to detect disease predisposition snps and high level interactions in genome wide association study,* BMC bioinformatics **10**, 1 (2009).

[23] P. Su, *Direct-to-consumer genetic testing: a comprehensive view,* The Yale journal of biology and medicine **86**, 359 (2013).

[24] A. M. Phillips, *Genomic privacy and direct-to-consumer genetics: Big consumer genetic data–what's in that contract?* in *Security and Privacy Workshops (SPW), 2015 IEEE* (IEEE, 2015) pp. 60–64.

[25] pgEd, *What is consumer genetics?* (2014), https://www.pged.org/direct-to-consumer-genetic-testing// Online; accessed November, 2018.

[26] O. Lukasz, K. Agnieszka, and C. Claude, *I?m 2.8% neanderthal,* 1st PETS Workshop on Genome Privacy (GenoPri) (2014).

[27] X. Jiang, Y. Zhao, X. Wang, B. Malin, S. Wang, L. Ohno-Machado, and H. Tang, *A community assessment of privacy preserving techniques for human genomes,* BMC medical informatics and decision making **14**, S1 (2014).

[28] L. Ricki, *Direct-to-consumer genetic testing: A new view.* (2012), http://blogs.plos.org/dnascience/2012/11/08/direct-to-consumer-genetic-testing-a-new-view/.

[29] P. J. McLaren, J. L. Raisaro, M. Aouri, M. Rotger, E. Ayday, I. Bartha, M. B. Delgado, Y. Vallet, H. F. Günthard, M. Cavassini, *et al.*, *Privacy-preserving genomic testing in the clinic: a model using hiv treatment,* Genetics in Medicine (2016).

[30] S. S. Shringarpure and C. D. Bustamante, *Privacy risks from genomic data-sharing beacons,* The American Journal of Human Genetics **97**, 631 (2015).

[31] K. Hamacher, *A taxonomy of genomic privacy and beyond,* 1st PETS Workshop on Genome Privacy (GenoPri) (2014).

[32] A. Ossola, *Gene tests are quite telling - should you get one?* (2015), http://www.popsci.com/i-tested-my-genes Online; accessed November, 2018.

[33] G. Danezis and E. De Cristofaro, *Simpler protocols for privacy-preserving disease susceptibility testing,* in *14th Privacy Enhancing Technologies Symposium, Workshop on Genome Privacy (GenoPri14), Amsterdam* (2014).

[34] 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* (ACM, 2013) pp. 95–106.

[35] Z. Brakerski and V. Vaikuntanathan, *Efficient fully homomorphic encryption from (standard) lwe,* SIAM Journal on Computing **43**, 831 (2014).

[36] R. L. Rivest, L. Adleman, and M. L. Dertouzos, *On data banks and privacy homomorphisms,* Foundations of secure computation **4**, 169 (1978).

[37] P. Paillier, *Public-key cryptosystems based on composite degree residuosity classes,* in *Advances in cryptology?EUROCRYPT?99* (Springer, 1999) pp. 223–238.

[38] C. Gentry, *Fully homomorphic encryption using ideal lattices,* in *Proceedings of the forty-first annual ACM symposium on Theory of computing* (2009) pp. 169–178.

[39] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, *(leveled) fully homomorphic encryption without bootstrapping,* in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (ACM, 2012) pp. 309–325.

[40] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig, *Improved security for a ring-based fully homomorphic encryption scheme,* in *IMA International Conference on Cryptography and Coding* (Springer, 2013) pp. 45–64.

[41] C. Ugwuoke, Z. Erkin, and R. L. Lagendijk, *Privacy-safe linkage analysis with homomorphic encryption,* in *Signal Processing Conference (EUSIPCO), 2017 25th European* (IEEE, 2017) pp. 961–965.

[42] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, *Manual for using homomorphic encryption for bioinformatics,* Proceedings of the IEEE **105**, 552 (2017).

[43] M. Namazi, J. R. Troncoso-Pastoriza, and F. Pérez-González, *Dynamic privacy-preserving genomic susceptibility testing,* in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security* (ACM, 2016) pp. 45–50.

[44] Z. Brakerski, *Fundamentals of fully homomorphic encryption - a survey,* Electronic Colloquium on Computational Complexity (2018).

[45] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, *A survey on homomorphic encryption schemes: Theory and implementation,* ACM Computing Surveys (CSUR) **51**, 79 (2018).

[46] O. Goldreich, *Secure multi-party computation,* Manuscript. Preliminary version , 86 (1998).

[47] R. Cramer, I. B. Damgård, *et al.*, *Secure multiparty computation* (Cambridge University Press, 2015).

[48] D. Giry, *Cryptographic key length recommendation,* (December, 2018), `https://www.keylength.com/en/4/` Online; accessed December, 2018.

[49] W. Du and M. J. Atallah, *Protocols for secure remote database access with approximate matching,* in *E-Commerce Security and Privacy* (Springer, 2001) pp. 87–111.

[50] Y. Lindell, *How to simulate it–a tutorial on the simulation proof technique,* in *Tutorials on the Foundations of Cryptography* (Springer, 2017) pp. 277–346.

[51] R. Canetti, *Universally composable security: A new paradigm for cryptographic protocols,* in *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on* (IEEE, 2001) pp. 136–145.

5

MACHINE LEARING ON PROTECTED GENOME DATA

Conducting outsourced machine learning on sensitive data posses a clear enough threat where there exist mistrusting parties. Therefore, the relevance of machine learning techniques encapsulated in privacy-preserving technologies are certain in an era enmeshed in multiple cloud services.

Parts of this chapter are under preparation:
(1) Ugwuoke, C., Toprakhisar D., Erkin, Z., Lagendijk, R. L, & Reinders M., (2020). REDACT: Machine Learning Model on Outsourced GWAS.
(2) Ugwuoke, C., Bliek, L., Erkin, Z., Veugen, T.,& Lagendijk, I. (2020). SCOTML: Collaborative Machine Learning Model for GWAS.

5.1. REDACT: MACHINE LEARNING MODEL ON OUTSOURCED GWAS

Genome-wide association studies (GWAS) is important to research and clinical medicine as it helps to identify associations between single-nucleotide polymorphisms (SNPs) and phenotypes or diseases. New SNPs are continuously discovered and tested for association to diseases and traits using regression models. Leading to new discoveries and better decision-making when treating patients. Due to computational intensive requirements of GWAS, it is common to outsource these computations to cloud services, hence the need to provide privacy for the outsourced data. Common privacy-preserving outsourced GWAS recommend multi-party computations or homomorphic encryption as privacy-protection techniques, and often require to compute one regression model per SNP. In this work, we propose REDACT, a privacy-preserving outsourced GWAS protocol. REDACT allows for homomorphically encrypted genome data to be outsourced to an untrusted third-party server for storage and to perform GWAS using logistic regression and compute p-values of multiple SNPs in parallel. We leverage the data packing technique in the CKKS homomorphic encryption scheme to obtain a parallelization algorithm thereby optimizing memory and computation resources. Given a dataset of 10643 SNPs, 245 individuals with 3 covariates, our prototype computes the logistic model and p-values under 24 hours on commodity hardware. Our algorithm offers up to 256 bits security against classical attacks, and is post-quantum secure for up to 192 bits security. An implementation of REDACT was submitted to the iDASH Privacy and Security 2018 Challenge. We propose one of the first parallelize-able outsourced solutions for securely computing regression models, and p-values over homomorphically encrypted dataset. Our protocol enhances GWAS outsourcing while protecting the privacy of collectively contributed genome data.

5.1.1. INTRODUCTION

Investigating the human genome can enhance researchers' understanding of complex relationships between diseases and genes, making it important to explore and ascertain any existence of such relationships [2, 3]. In certain cases, a disease of interest may point to numerous locations on the genome as possible candidates for "causative genes" [4]. The study of the associations between genes and phenotype such as diseases and traits is classified as genome-wide association study (GWAS) [4, 5]. In order to carry out such studies, GWAS commonly depend on single-nucleotide polymorphisms (SNPs) as primary dataset. Nucleotides are considered the building blocks for deoxyribonucleic acid (DNA), and can take any of 4 possible bases namely (A, T, C, G). It is common that there exists a genetic variation between individuals of the same species, and this variation could happen as a result of a single substitution of a nucleotide base when compared to a common reference genome [6]. A single variation in the nucleotide is classified as a SNP if it is not observed in more than 1% of the population [7]. A SNP may influence an individual's risk of having a particular disease, thus, it makes it germane and interesting for researchers to conduct GWAS in order to establish existence of any such association [1, 2, 5, 8, 9]. For example, a GWAS analysis that seeks possible association between a SNP and a disease of interest, would typically involve hundreds of individuals and thousands of candidate SNPs [1]. These individuals are generally classified into two groups named *case* and *control*, with additional covariates such as gender, age, weight, etc. Whereas the *case* group contains individuals whom are known to have the disease, individuals in the *control* group are free from the disease. Thereafter, a statistical model is developed for each of the SNPs and the significance of each SNP computed.

Simply put, given a dataset containing hundreds of individuals classified into the *case* and *control* groups for a disease of interest, and a SNP of interest, one can generate a regression model that can estimate the significance of that SNP to the disease. In order to compute the statistical tests for association given an individual's SNPs, basic regression models such as linear regression and logistic regression suffice and are commonly deployed [1]. Although, where it is the case that the outcome of the model is binary as with disease or no-disease, logistic regression model is preferred due to its dichotomous outcome. After a model is obtained, the SNPs are tested for significance by computing corresponding *p-values* over the model parameters. Computing *p-values* for SNPs is used in classifying the SNPs into two pools, significant and non-significant SNPs. Thereby narrowing further advance search of disease-gene relationship through expensive experiments to only the significant SNPs.

Increase in the size of learning dataset is directly proportional to improving the prediction accuracy of many learning algorithms of which logistic regression is one [14–16]. In our GWAS scenario, dataset increase is equivalent to increasing the number of individuals in both the *case* and *control* group. However, it is not commonly the case that a single institution such as a research institute, hospital, or pharmaceutical company would have all the data required for training a machine learning a model. Especially where the disease, which happens to serve as the dependent variable, is considered rare. Therefore, in order to improve the accuracy of a model and reduce bias, it becomes necessary to share data amongst institutions, especially for rare diseases or rare SNPs. In situations where the model to be computed is considered complex and may require multiple de-

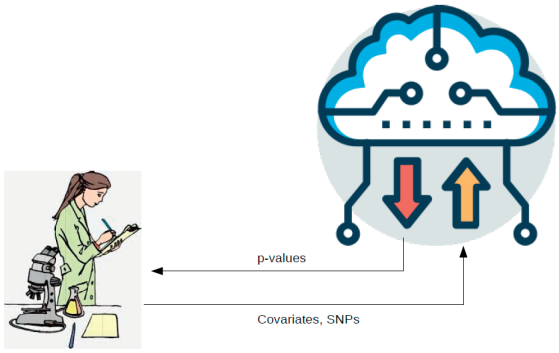

Figure 5.1: Protocol Setting for REDACT

pendent variables with thousands of records, the task becomes computational expensive for commodity hardware [17]. Consequently, in order to circumvent the computational drawback, researcher can outsource the computation of the model and other relatively expensive computations to a third-party server. The third-party server can be considered to be a cloud services provider with sufficient computational and storage resources, and is responsible for computing the learning model using any submitted dataset. For this reason, outsourcing genome data for scenarios such as described above is not only relevant, but practicable and commonly adopted as shown in Figure 5.1.

SNPs are highly sensitive dataset and as such must be protected [10–12, 18–20]. Leaking only a few hundred SNPs can already pose significant privacy-risks to the owners. Making it unsuitable to share genome data with a third party in an unprotected format. Because genome data are considered to have severe privacy-risk, even though it is needed for outsourcing to a mistrusting third party, there exist an intrinsic need for privacy-protection that is commensurate with the privacy-sensitivity of the data. In an attempt to proffer solutions to the privacy requirements posed by outsourcing genome data to third party services for the aim of carrying out GWAS, researchers have adopted various cryptographic techniques. The most dominant cryptographic solution has been homomorphic encryption based solutions [10–13]. However, existing solutions commonly lack either or both of two important properties required for outsource GWAS, namely *non-interactivity* and *parallelization*. The non-interactive property requires that once data is outsourced to the third party service, the third party should not have to interact with any other entity in order to complete the computation. This means the third party service should only accept input data from the data contributor or owner, and should output the final result in a privacy-protected format. Fully homomorphic encryption schemes are commonly adopted to provide this property. The demand for parallelization property is usually associated with the success in the former property. When non-interactivity is achieved, it usually comes at high computation cost attributed to homomorphic operations. Hence the need to parallelize the computation in order to optimize computation and memory resources.

In this work, we present REDACT, a parallelized, non-interactive, homomorphic encryption based approach that allows SNPs to be outsourced to a mistrusting third party

service for the purpose of computing GWAS. The third party receives homomorphically encrypted SNPs and relevant covariates, with which it generates a logistic regression model for every SNP. Then, computes the significance of each SNP by way of computing *p-value*. And finally returns encrypted *p-values* as the final output. REDACT achieves both *non-interactivity* and *parallelization* properties, thereby making it one of the first solutions to achieve these feat using encrypted data.

REDACT is presented in a semi-honest security setting, where all parties are expected to follow the rules of the protocol. The security and privacy of REDACT is intrinsic in the homomorphic encryption scheme adopted, by that, making the proof deductive. Hence, we will not provide any formal proof for the protocol, but rather refer the reader to the proof of the adopted encryption scheme.

Our contributions are as follows:

- Given encrypted genome dataset, we propose a privacy-preserving *Machine Learning As a Service* (MLAS) protocol, where users can outsource machine learning model generation to an untrusted third party server.
- Data classification of subsequent encrypted dataset can be performed using the encrypted models generated from above.
- Test statistics can be performed by the third party server on the encrypted machine learning model, and an encrypted test statistic is generated for each SNP/model in the genome dataset.
- The 3 phases listed above can be parallelized to generate up to 2048 models and their corresponding *p-values* in parallel.
- Our results show that with a dataset of 10,643 SNPs, 245 individuals each with 3 covariates, which was provided for the iDASH 2018 Challenge, our protocol can generate the logistic regression models and the *p-values* in about 23 hours, using commodity hardware.

The remainder of this paper is structured as follows: In Section 5.1.2 we provide relevant literature to our proposal. Section 5.1.3 provides preliminary building blocks relevant to our solution. In Section 4, we present REDACT. Section 5 contains implementation details. In Section 6 we present the results. Finally we conclude in Section 7.

5.1.2. RELATED WORKS

Kim et al. [10] proposed a secure and privacy-preserving logistic regression outsourcing protocol which takes homomorphically encrypted data as input, and generates a homomorphically encrypted model for classification. Although their work was a breakthrough for non-interactively learning a regression model using encrypted GWAS data, it was not designed to support computation of *p-values* for the generated models. As such, their solution aimed at providing privacy protection using homomorphic encryption which allows a data owner to outsource encrypted data to a third party server, who is then required to generate learning models using logistic regression. Also, Kim et al's solution only provides for a single SNP model, which implies that if multiple models were to be

required for different SNPs, then the algorithm would compute the models by mean of reiteration. Although Kim et al. harnesses the packing and parallelization techniques inherent in CKKS scheme [21], it was not targeted to parallelize for multiple models.

Gildad-Bachrach et al. [22] proposed CryptoNets, which demonstrates how neural networks can be trained over encrypted data, using homomorphic encryption schemes. Thereby providing privacy and confidentiality to outsourced data, while being able to enjoy the benefits of the neural network. CryptoNets only provides data classification function using homomorphically encrypted dataset but does not generate the machine learning models over encrypted dataset. The classification phase is however parallelized in CryptoNets, although that does not compare to the complexity of parallelizing encrypted model generation.

Just like Kim et al. [10], Bonte et al. [16] independently proposed a solution to computing privacy-preserving logistic regression over homomorphically encrypted data. Their solution is equally limited to iterative computation of the classification model where multiple models are required. Generally, computing classification models over encrypted dataset is usually associated with huge computational complexity, as evident in the works above. More so, when multiple models are required, repetitive algorithms only scale up the complexity, hence the need for algorithms that are parallelize-able.

In order to reduce the huge computation complexity associated with iterative computation of learning models where multiple SNPs are targeted, Sikorska et al. [1] introduced a novel approach to parallelize the computation of the model as well as their associating *p-values*. However, the approach does not envisage the outsourced data as privacy-sensitive, hence it does not consider any privacy-protection to the algorithm. While Sikorska et al. [1] was able to eliminate the complexity of the outsourced computation, the privacy concerns associated with sharing GWAS dataset remains open. Similarly, Buzdugan et al. [23] equally present a solution that allows GWAS data to be analyzed in a parallelized manner, allowing for multiple models to be computed in parallel, as well as corresponding *p-values* for the models.

In this paper, in an attempt to aggregate the positives in the above listed papers, we present REDACT as a privacy-preserving logistic regression outsourcing protocol, which implements the parallelization techniques proposed by Sikoska et al. [1] and preserves the accuracy of the encrypted results.

5.1.3. PRELIMINARIES

Here, we introduce building blocks that are utilized to realize REDACT. We offer the details that are relevant to the context of or problem setting and the solutions we adopt. These building blocks will include genome dataset and their structures, the basic machine learning algorithm that is chosen for training and classification, and finally the cryptographic primitives that are required to achieve the privacy and security goals.

SINGLE NUCLEOTIDE POLYMORPHISM

A SNP is considered to have occurred at a locus i of an individual's genome, if there is a substitution of a nucleotide at locus i. In such a condition, the individual is said to have $SNP_i = j$, where $j \in \{0,1\}$. The value of j denotes the presence or absence of a SNP, represented with either 1 or 0 respectively. Consider Table 5.1, individuals "hu6E45" and

"hu34D5" have SNP values 1 at locus "chr1_44" while "hu4386" has SNP value 0 at the same locus.

ID	chr1_33	chr1_44	chr1_55	chr1_66
hu4386	1	0	0	1
hu6E45	0	1	0	0
hu34D5	0	1	1	0

Table 5.1: Human SNP Data

LOGISTIC REGRESSION

Logistic regression is a mathematical modeling approach utilized in describing the relationships of multiple independent variables to a dichotomous depend variable [24]. Logistic regression as a machine learning algorithm is predominantly used in epidemiology community [25]. It is suitable for cases with two outcome classification. The dependent variables are commonly dichotomous, with discrete values in a simple set such as {"1", "0"} . The independent variables or covariates can be continuous and can number as many as possible. Eq. 5.1 is an example of a classical logistic regression model, with n covariates, where $p := Pr(Y = 1|x)$, and Y being the response variable.

$$logit(p) = \ln(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n. \tag{5.1}$$

From Eq. 5.1,

$$\frac{p}{1-p} = e^{(\beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n)} \tag{5.2}$$

From Eq. 5.2, X is a matrix of the independent variables or covariates, containing m individuals. While β is a vector of the model parameters which are to be estimated, otherwise known as coefficient of the covariates.

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \ldots & x_{1,n} \\ x_{2,1} & x_{2,2} & \vdots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m,1} & x_{m,2} & \ldots & x_{m,n} \end{bmatrix}, \ \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix} \tag{5.3}$$

Let α_i denote the logit function, and defined as:

$$\alpha_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n.$$

$$\alpha = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \ldots & x_{1,n} \\ 1 & x_{2,1} & x_{2,2} & \vdots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & x_{m,1} & x_{m,2} & \ldots & x_{m,n} \end{bmatrix} \cdot \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix} \tag{5.4}$$

$$p = \sigma(\mathbf{X};\beta) := \frac{1}{1+e^{-\alpha_i}} \tag{5.5}$$

Figure 5.2 represents a graph of a standard logistic regression function.

From Eq. 5.5, given a sample dataset containing independent variables X and their corresponding dependent variables Y , the vector β is computed and used as input for classifying/predicting the Y values for a set of new X values. The process of estimating the β values is termed maximum likelihood estimation (MLE). MLE can be computed in more than one way, it is often important to consider the most efficient method based on the structure of the dataset.

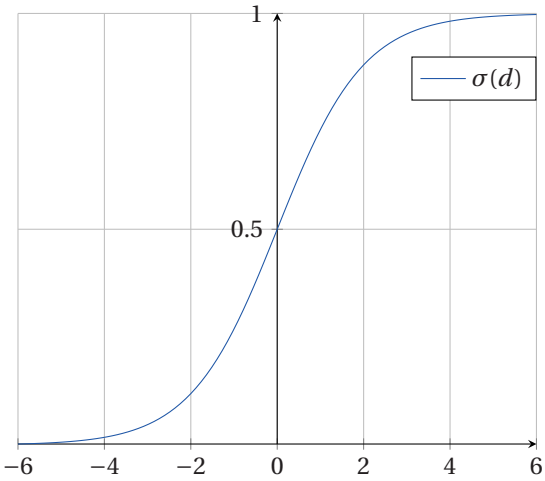

Figure 5.2: A sigmoid function graph.

MAXIMUM LIKELIHOOD ESTIMATION

The idea here is to maximize a log-likelihood function such as the probability of success, by estimating values for the parameter β . Consider Eq. 5.5, for a given vector X, the value of p can be optimized for some values of the parameter β . Commonly used methods include gradient descent, Newton-Raphson, Fisher Scoring, and *Iterative Reweighted Least Squares (IRLS)* [25]. All four methods require some levels of iterations to optimize β . While gradient descent method is relatively easier to compute due to its use of basic mathematical operations, it however requires more numbers of iterations that its other counterparts. The other methods generally depend on matrix operation, including a matrix inversion, this makes them relatively complex when compared to gradient descent approach. However, they require very few number of iterations when compared to gradient descent. The general equation of IRLS can take the form:

$$\beta^{(t+1)} = (\mathbf{X}^{T}\mathbf{W}^{(t)}\mathbf{X})^{-1}\mathbf{X}^{T}\mathbf{W}^{(t)}\mathbf{Z}^{(t)} \tag{5.6}$$

Where t represents the iteration step, X^T is the transpose of X, and $W^{(t)}$ is the inverse of the covariance matrix with diagonal values as $p_i^{(t)}(1-p_i^{(t)})$.

The reader can refer to [25] for further details.

HOMOMORPHIC ENCRYPTION

A homomorphic encryption (HE) scheme allows for arbitrary algebraic operations to be performed on ciphertexts. Let $Enc_{pk}(\cdot)$ and $Dec_{sk}(\cdot)$ represent encryption and decryption functions respectively. (m_1, m_2) are two messages and k is a scalar value, while $\boxplus$, $\odot$ and $\boxtimes$ are arbitrary operations on the ciphertexts. Then, homomorphism is defined as follows:

$$Dec_{sk}(Enc_{pk}(m_1) \boxplus Enc_{pk}(m_2)) = m_1 + m_2, \tag{5.7}$$

$$Dec_{sk}(Enc_{pk}(m_1) \odot Enc_{pk}(m_2)) = m_1 \cdot m_2, \tag{5.8}$$

$$Dec_{sk}(Enc_{pk}(m_1) \boxtimes k) = m_1 \cdot k. \tag{5.9}$$

HE schemes can further be classified into two types: partial HE schemes and fully/levelled HE schemes. Fully/levelled HE schemes allow for addition and multiplication operations to be carried out on ciphertexts, while partial schemes often allows either the addition or the multiplication operation [26]. In this work we desire the properties of the fully/levelled HE scheme, and we will continue with the rest of the paper focusing on fully/levelled homomorphic encryption (FHE) schemes.

HOMOMORPHIC ENCRYPTION LIBRARIES

A number of standardized libraries implement fully homomorphic encryption (FHE) schemes, and they usually differ in performances, key sizes and properties, to mention a few. However, the choice of library is often a function of the algorithm to be implemented because different libraries offer varying functionalities making them suitable for one scenario but not so optimal in another scenario. In Table 5.2, we present the properties and functionalities that guided our choice of library, at the time of implementing.

	HELib	SEAL 2.0	HEAAN	Palisade
Decimal Support	No	Yes	Yes	No
Matrix Operations.	No	No	Yes	Yes
Parameter Choice	Easy	Easy	Easy	Difficult
Rotation Operation	Easy	Easy	Easy	Difficult
Easy Noise Mgt.	No	Yes	Yes	No

Table 5.2: FHE libraries and properties

HEAAN

A leveled HE scheme named homomorphic encryption for arithmetic of approximate numbers (HEAAN) was introduced by Cheon et al. [21], and it implements the FHE scheme proposed by Brakerski et al [27]. HEAAN is designed based on the ring learning with error (RLWE) problem, and approximates results of homomorphic evaluations by allowing small errors. A summary of the scheme is presented below:

Let $\mathscr{R} = \mathbb{Z}[X]/(X^N + 1)$ represent a cyclotomic ring of integers, where $N = 2^k$, being the dimension of the ring and k is a positive integer.
$\mathbf{R}_q = \mathbf{R}(\mod q) = \mathbb{Z}_q[X]/(X^N + 1)$, where q is the ciphertext modulus. For a fixed base $p > 0$ and a modulus q_0, $q_\ell = p^\ell \cdot q_0$ for $0 < \ell \leq L$, and L being the maximum level of the ciphertext. A ciphertext $\mathbf{c}$ of level ℓ is a vector in $\mathbf{R}_{q_\ell}^k$ for a fixed integer k. Let λ represent the security parameter.

- **KeyGen**(1^λ) :
  For a chosen ciphertext modulus level L, and given security parameter λ, output N being the dimension of the ring.

  Set the distributions $\chi_{key}, \chi_{enc}, \chi_{err} \in \mathscr{R}$ to represent the secret key, encryption and error, respectively.

  Sample $s \leftarrow \chi_{key}$, $e \leftarrow \chi_{err}$, $a \leftarrow \mathbf{R}_{q_L}$. Set the secret key as $sk \leftarrow (1, s)$, and public key $pk \leftarrow (b, a) \in \mathbf{R}_{q_L}^2$, where $b \leftarrow -a \cdot s + e \ (\mod q_L)$.

- **Enc**$_{pk}(m)$:
  Given a polynomial $m \in \mathbf{R}$, Output a ciphertext $\mathbf{c} \in \mathbf{R}_{q_L}^k$, such that $\langle \mathbf{c}, sk \rangle = m + e \ (\mod q_L)$.

- **Dec**$_{sk}(\mathbf{c})$:
  For ciphertext $\mathbf{c}$ at level ℓ, output a polynomial $m' \leftarrow \langle \mathbf{c}, sk \rangle (\mod q_l)$, for the secret key sk.

- **Add**$(\mathbf{c_0}, \mathbf{c_1})$:
  Given two ciphertexts of the polynomials m_0, m_1 as $\mathbf{c_0}, \mathbf{c_1}$, output a ciphertext $\mathbf{c}_{Add}$, such that $(m_0 + m_1) \leftarrow \langle \mathbf{c}_{Add}, sk \rangle (\mod q_l)$.

- **Mult**$(\mathbf{c_0}, \mathbf{c_1})$:
  Given two ciphertexts $(\mathbf{c_0}, \mathbf{c_1})$, output a ciphertext $\mathbf{c}_{mult} \in \mathbf{R}_{q_\ell}^k$, such that $\langle \mathbf{c}_{mult}, sk \rangle = \langle \mathbf{c}_0, sk \rangle \cdot \langle \mathbf{c}_1, sk \rangle + e_{mult} (\mod q_\ell)$, where $e_{mult} \in \mathbf{R}$.

- **RS**$_{\ell \rightarrow \ell'}(\mathbf{c})$:
  For a ciphertext $\mathbf{c} \in \mathbf{R}_{q_\ell}^k$ at level ℓ and a lower level $\ell' < \ell$, output a new ciphertext $\mathbf{c}' \leftarrow \lfloor \frac{q_{\ell'}}{q_\ell} \rfloor \in \mathbf{R}_{q_\ell}^k$.

NOTATION

In Table A.1, we provide relevant notations and their explanations as used in the rest of this paper.

Table 5.3: Notations

Notation	Description
$\hat{a}$	The hat makes it the estimated value of a
$\mathbf{X}$	Variables in bold represent vectors or Matrices
$\mathbf{X}^T$	Transpose of the matrix $\mathbf{X}$
ℓ	Multiplication depth of the circuit/protocol
$logp$	Plaintext modulus, which determines the precision of the plaintext
$logq$	Minimum ciphertext modulus for decryption to be feasible
$logQ$	Initial ciphertext modulus which compensates for multiplication depth.
λ	Security parameter, with default value of 256-bit security.
$slots$	Number of slots allocated for packing ciphertexts.
$cohotSize$	Number records/individual in the dataset.
$sizeSNP$	Number of SNPs contributed by each record/individual.
y	Response variable for the presence or absence of phenotype.

5.1.4. REDACT

In this section, we provide the setting of our problem and a detailed description of our protocol.

PROTOCOL SETTING

The problem of computing *p-values* for a given SNPs dataset can be solved by training logistic regression in two phases on the given dataset. We adopt the two model approach, which entails to first compute a base model over the covariates (excluding the SNPs). Secondly, with the base model, the SNPs are subsequently introduced using the Ralphson-Newton algorithm.The corresponding coefficient of the individual SNPs are then computed to obtain a second model. For our challenge, we are provided with a dataset that contains the following:

- 245 individuals who are either classified into case or control class for a particular disease.
- covariates include (age, height, weight).
- 10643 SNPs are provided for each individual.

REDACT DESIGN

The basic idea in this problem is to take as input encrypted SNPs and covariates, and from that compute the corresponding *p-values* for every SNP.

The procedure is roughly as follows:

- Compute values for p from Eq. 5.5.
- Compute values for w from p values.
- Compute $z = \log\left(\frac{p}{1-p}\right) + \frac{y-p}{w}$.

- Compute $s^* = s - X(X^T W X)^{-1} X^T W s$.
- Compute $z^* = s - X(X^T W X)^{-1} X^T W z$.
- $\hat{\beta}_1 = \frac{\sum_i w_i z_i^* s_i^*}{\sum_i w_i s_i^{*2}}$,
- $\hat{var}(\beta_1) = \frac{1}{\sum_i w_i s_i^{*2}}$.
- $err = SQRT(\hat{var}(\beta_1))$.
- $p\text{-}value = 2 * \text{pnorm}(\beta_1 / err)$.

Algorithm 2 REDACT

procedure INITIALIZATION
  Set variable N, $\log P$, $\log Q$, $logSlots$, $cohortSize$, $sizeSNP$, $slots$.
  Encrypt SNP values in packs of size $slots$.
  Encrypt each covariate values in packs of size $slots$.
  Send all ciphertext to storage and processing unit.
end procedure
procedure Storage_and_Processing_Unit
  $a[4] = [0.49989432, 0.24486503, 0, -0.01414489]$.
  for $i = 0 \rightarrow (cohortSize - 1)$ **do**
    $\hat{p} = a_0 + \beta(a_1 + a_3 * \beta^2)$
    $w = \hat{p} * (1 - \hat{p})$.
    $z = log(\hat{p}/(1-\hat{p})) + (y - \hat{p})/(\hat{p} * (1 - \hat{p}))$.
  end for
  $xtw = (X * w)^T$.
  $U_1 = X^T W * z$.
  $U_2 = linearEquation(xtw * X, U_1)$.
  $z^* = z - X * U_2$.
  $U_3 = xtw * S$.
  $U_4 = linearEquation(xtw * X, U_3)$.
  $S^* = S - X * U_4$.
  $S^{*2} = sum(W * S^{*2})$.
  $\beta = crossProduct(Z^* * W, S^*)/S^*$.
  $err = SQRT(1/S^*)$.
  $p\text{-}value = 2 * pnorm(-abs(\beta/err))$.
end procedure
Return *p-value*

All data are encrypted using a fully homomorphic encryption (FHE) scheme, and the encrypted data is transferred to a third party storage and processing unit for the purpose of computing p-values of each of the 10643 SNPs, and return an encrypted dataset. We adopt the Homomorphic Encryption for Arithmetic of Approximate Numbers [21]

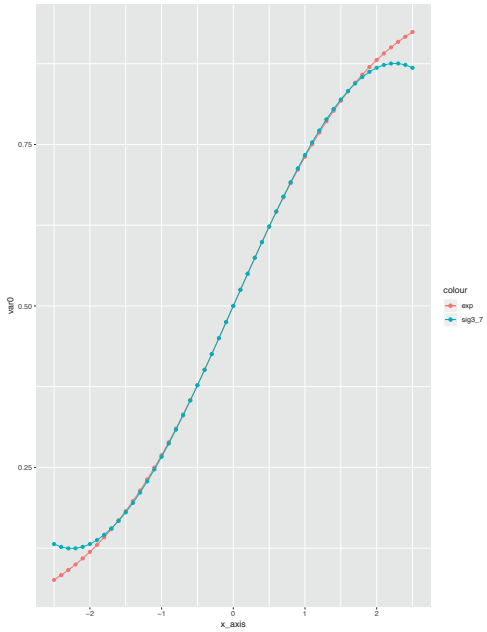

Figure 5.3: Approximated Sigmoid function (sig3_7) against the true Sigmoid function (exp).

library, which implements the FHE scheme proposed by Brakerski et al [27]. Leveraging on the properties of HEAAN, and for our chosen parameters, we could pack up to 2048 SNPs into a single ciphertext. This packing technique enhances our storage, memory and processing optimization, making it easy to process a packed dataset therefore achieving parallelization, as against processing individual data entry.

Using the packed ciphertext, a logistic regression base model is created using the covariates excluding SNPs. Afterwards, the coefficients of the base model are estimated with log likelihood function. This base model is followed by an updated model using the weights from base model, for each pack of SNPs, coefficients of the SNPs are estimated and with the errors of these coefficients, Z Statistic are computed for each SNP in the dataset. Following the same technique as proposed by Sikorska et al [1], we implement the Newton-Ralphson algorithm to obtain coefficients of the model[28]. The Newton-Ralphson technique is iterative and requires matrix inversion. For that, we decompose the matrix inversion into a somewhat complex simultaneous equation and solve for the inverse, in order to continue with each iteration. All our computations are done over encrypted data, using HEAAN library[21] and C++.

IMPLEMENTATION

Challenges encountered during the implementation of REDACT, and corresponding solutions include:

- **Sigmoid function:** we use approximation function for which the coefficients are obtained using gradient descent, see Figure **??**. Sigmoid function approximation

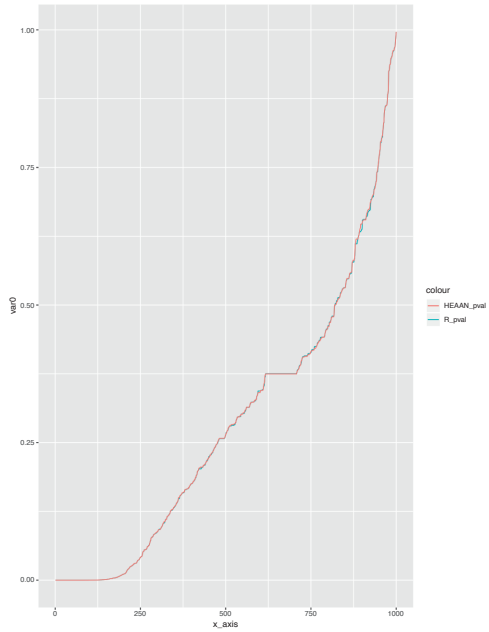

Figure 5.4: *p-values* from REDACT compared with glm variant.

uses the equation:

$$\hat{p} = 0.49989432 + x(0.24486503 - 0.01414489x^2)\,.$$

The range of $x \in \{-2\,,2\}$ was determined by observing the data in plaintext, which means that a pre-knowledge of values in x is required for using REDACT.

- **Matrix inversion:** we designed a somewhat simultaneous equation where we decompose elements of the matrix into coefficients of a simultaneous equation, and proceed to solve same equation to obtain the inverse matrix. Note that this is done using all encrypted inputs and computed non-interactively. See the $linearEquation()$ function in the source code for details.
- **Division:** approximation using existing function provided in HEAAN library.
- **Logarithm:** approximation using existing function provide in HEAAN library.
- **Matrix multiplication:** we designed a function to handle matrix multiplication with packed encrypted values. The idea here is to introduce data packing technique in order to parallelize the computation. See the $matMultV()$ function in the source code for details.
- **Square root:** we adopt the Babylonian method for approximating square root function. $x_{n+1} = 0.5 * (x_n + \frac{y}{x_n})$, where x_{n+1} is the square root for a positive integer y .

This method approximates the value of y using homomorphically encrypted x values. The homomorphically estimated *p-values* can be seen plotted against those computed in plaintext in Figure 5.4. See the $ZTest()$ function in the source code for details.

Parameters for the homomorphic encryption scheme:

- Multiplication depth : 110
- logN = 12
- precision bits $logp = 28$
- initial ciphertext modulus $logQ = 3143$
- final ciphertext modulus $logq = 63$.

5.1.5. SECURITY AND PRIVACY

All dataset outsourced to the storage and processing unit are encrypted using the fully-homomorphic encryption scheme. And all operations and analyses are performed over the encrypted dataset. Therefore, we argue that the privacy and security of the data is as secure as the encryption scheme and parameters chosen.Our algorithm offers up to 256 bits security against classical attacks, and is post-quantum secure for up to 192 bits security.

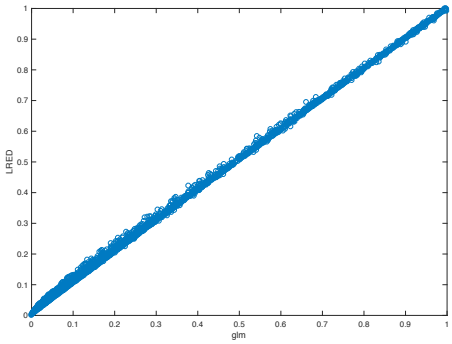

Figure 5.5: Scatter plot of *p-values* obtained from REDACT(LRED) against glm.

5.1.6. RESULTS

We developed our logistic regression approximation algorithm using the provided dataset. This dataset have single binary outcome variable, which informed our decision to train using logistic regression as binary classifier. From the obtained classifiers, we homomorphically compute the Z Statistic of the provided SNPs. For the given dataset, our initial solution requires about 6.4 GB of storage space, less than 6 GB of RAM and takes approximately 20 hours to complete on a commodity hardware. In Figure 5.5, we present a scatter plot of *p-values* obtained using REDACT, and computed using an R package

called “glm”. The plot indicate that REDACT offers as much as 95 % accuracy when compared to the non-privacy-preserving approach obtained using glm.

5.1.7. CONCLUSIONS

The objective of this work is twofold. First is to solve the underlying problem of security and privacy when outsourcing genome data processing. With REDACT, we provide evidence on how fully-homomorphically encrypted data can be outsourced to a semi-honest storage and processing unit, and non-trivial machine learning computations can be accurately computed and results returned to the client. Second, is to present a proof of concept of how primitive mathematical operations can be realised over encrypted dataset, in a non-interactive setting. We achieve both objectives while still being able to provide up to 95% accuracy and a 256-bit security against known classical attacks, while providing up to 192-bit security against known post-quantum attacks.

5.2. SCOTML: COLLABORATIVE MACHINE LEARNING MODEL FOR GWAS

Digital copy of the human genome is a rich dataset and is considered privacy-sensitive, which must be protected. Conducting machine learning (ML) on genome dataset is an important aspect of research and clinical medicine and often leads to new discovery and better decision-making when treating patients. Collaborations between research centres and medical institutions are common practice while attempting to learn from existing genome dataset, and they can be mistrusting entities. Therefore, such collaborations must be done without compromising the privacy of the genome data used for training the machine learning model. This, reinforces the need for a secure and privacy-preserving protocol for collaborative machine learning models. Existing privacy solutions for collaboratively training a ML model includes techniques such as homomorphic encryption, which can be computationally expensive. In this work, we propose SCOTML, a privacy-preserving machine learning protocol which allows 3 or more mistrusting parties to jointly train a machine learning model with their private genome dataset. SCOTML was submitted to Track 4 of the iDASH 2019 competition. We adopt random features expansion (RFE) as our learning algorithm, and implement a secure multiparty computation variant that allows 3 parties to jointly train a ML model. We implement our protocol in Python3 and provide a security argument using the simulation based security proof. Our implementation offers up to 99.8% accuracy on the "BC-TCGA" dataset and 73% accuracy on the "GSE2034" dataset, and completes training in less than 40 seconds on commodity hardware.

5.2.1. INTRODUCTION

The availability of cheap genome sequencing in recent years has transformed the digital genome processing landscape [29], thereby resulting to more awareness, and evolving needs for processing digital genomes. *In silico* genome has become more affordable thereby leading to a trove of data being made available to individuals, research community, and commercial companies [30]. The reason for sourcing a digital genome varies from one party to another simply because information inherent in the genome is such that multiple players could explore different pieces of the entire genome. Whatever the reason for seeking genome data processing is, these reasons are often interconnected and one phase of genome data processing life cycle only helps to optimise the next phase.

When people adopt some services associated with genome data processing, they commonly seek to have their genome sequence in pursuit of some personal interest such as ancestry information, paternity or maternity tests, predisposition to diseases. Some of these test are only possible due to the existence of results from previous studies, which are then used to decide threshold for making decisions in the current computation, making them interdependent as seen in Fig. 5.2.1.

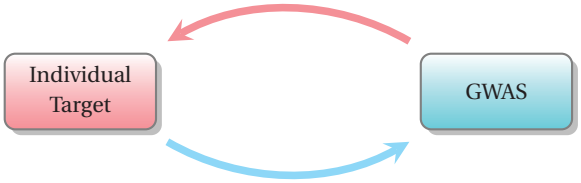

Figure 5.6: Phase relationship of genome data processing

For instance, genome wide association study (GWAS) mainly focuses on a population study, where information about a gene phenotype relationship is investigated using multiple subject data [31, 32]. Which means that individuals must be willing to provide their data in order to improve the GWAS computation phase, and from the improved GWAS results, other individuals benefit by having more accurate results when they utilize GWAS generated outputs. GWAS is mostly of interest to the research community and commercial companies, while its results are used to service individual needs. For example, results of a GWAS is utilized by individuals who might be interested in learning their predispositions to phenotypes of interest. This means that individuals can learn from GWAS while being able to contribute their data in order to improve prediction accuracy of the GWAS models.

GWAS is hugely reliant on machine learning (ML) algorithms in order to build the models used for classification and in extension, predictions. ML has also found applications in other aspects of genomics including variant calling and pathogenicity scores [30, 33, 34]. Just like every machine learning problem, more data usually results in better models being generated. The ML algorithm to be adopted in any GWAS varies with the needs and properties of both data and algorithm, hence there are multiple implementations of algorithms in solving GWAS related problems. Common algorithms include linear regression, logistic regression, deep learning [33, 35, 36].

We use the term “institution" loosely to represent either a research centre or a commercial company. The need for richer dataset with which to perform GWAS-related machine learning computations, has made it necessary for multiple institutions to seek collaborations in order to improve their machine learning models. However, the concern for the privacy-sensitivity of genomics data makes the data sharing a non-trivial problem. The question then becomes: in the face of privacy concerns, can multiple parties securely collaborate to jointly generate a public machine learning model while protecting the privacy of their contributing dataset?

Secure collaborative learning has remained an open and interesting problem, attracting various solutions from the research community [37]. Existing solutions include the use of encrypted data, data locality, and secure multiparty computation [38–40] techniques to ensure that their contributing data is protected from the prying eyes of potential adversaries. A common drawback to the existing secure solutions can be grouped into three parts namely: accuracy of model, speed of computation, and communication cost. It is important to mention that the security/privacy technique is often determined by the setting and scenario of the problem. For example, where the computation is required to be done on a central processing point such as the cloud, technique such as homomorphic encryption becomes preferable, due to the non-interactivity proper it provides. Contrary, where data is distributed without a centrally trusted party, technique such as secure multiparty computation becomes more suitable.

The focus of this work is to answer the above question in a semi-honest security setting where three or more mistrusting institutions could securely merge their private data for the computation of a public machine learning model, and provide a proof of concept with provable security argument to support the proposed solution. Our proposal for a secure collaborative training of machine learning model (SCOTML) was submitted to the Track IV of the *integrating data for analysis, anonymization, and sharing* (iDASH) 2019 competition.

In support of the research toward promoting privacy for genome data, iDASH [41], whose focus is on privacy-preserving algorithms and solutions for data sharing, have continued to push the boundaries of unresolved problems in genome data processing research and algorithms. One way iDASH does this is by announcing open problems as challenges yearly, and accepting contributions from all over the world in hope of closing some research gaps, solving previously open problem, and realizing more secure, privacy-preserving, and efficient solutions to genome data processing.

In this paper, we present the following contributions:

- We propose SCOTML to bridge the research gap by providing privacy, high accuracy, and high efficiency for three or more mistrusting parties to collaboratively learn from a high dimensional dataset in an honest-but-curious security setting.
- Our prototype implementation of SCOTML is relatively fast and can be be completed on commodity hardware in about 53 seconds even for a dataset with 472 individuals and each with 17,814 genes of interest.
- To the best of our knowledge, we provide the first privacy-preserving protocol design that utilizes Random Features Expansion for learning on collaborative sourced dataset.

Our implementation proves to be secure and rival their unsecured variants both in accuracy and speed of computation.

5.2.2. RELATED WORKS

Relevant literature in security, machine learning and bioinformatics are to be discussed with their relationship to our work spelt out.

Although machine learning techniques have been quite commonly adopted in the field of bioinformatics [1, 42–45], research contributions towards privacy-preserving machine learning have equally developed over the years [46–48]. The privacy-preserving machine learning solutions that have been proposed towards the protection of genome dataset are equally common in literature, and they have been designed with the use of variety of machine learning algorithms. Usually, the problem definition and machine learning algorithm determines to a large extent the privacy-preserving method suitable.

For instance in [49], Kim et al. demonstrate with the use of homomorphic encryption technique how a privacy-preserving machine learning can be achieved for genome data. Kim et al. choose logistic regression as the machine learning algorithm, and their results show that both the learning phase, and prediction phase of the machine learning process can be achieve with relative accuracy between 61% and 91% depending on the dataset used. These results were achieve using all encrypted data outsourced to cloud infrastructure non-interactively, hence the drawback in efficiency with some algorithms taking as much 235 minutes to complete.

In the efforts to improve the proposal in [49], Choen et al. proposed another non-interactive machine learning solution which used logistic regression to process homomorphically encrypted genome data [50]. This proposal introduces new and efficient techniques for parallelizing the process, and for computing gradient descent which contributed most of the computational overhead in [49]. While these approaches solve privacy challenges inherent in outsourcing genome data processing with the use of machine learning algorithms, they do not suffice for collaborative data processing settings.

Other machine learning applications towards the protection of sensitive data adopt ensemble approach by aggregating the prediction from multiple models [51–53]. Although ensemble techniques attempts to preserve the privacy of contributed data used for machine learning models generation in a collaborative setting, they do not account for scenarios where the data need to be collated and processed in order to preserve structures and improve accuracy of the model. Ideally, we would want that all the data are combined and processed using the same algorithm while preserving the privacy of the data.

5.2.3. BACKGROUND

We introduce in detail the various techniques that are relevant to solving the stated problem. These include all building block that aids the reader to comprehend each component of our solution and the complexity thereof.

5.2.4. COLLABORATIVE MACHINE LEARNING ON GENOME DATA

We consider a supervised learning problem where the goal is to detect breast cancer in patients based on their genome data as described in the iDASH 2019 competition [54].

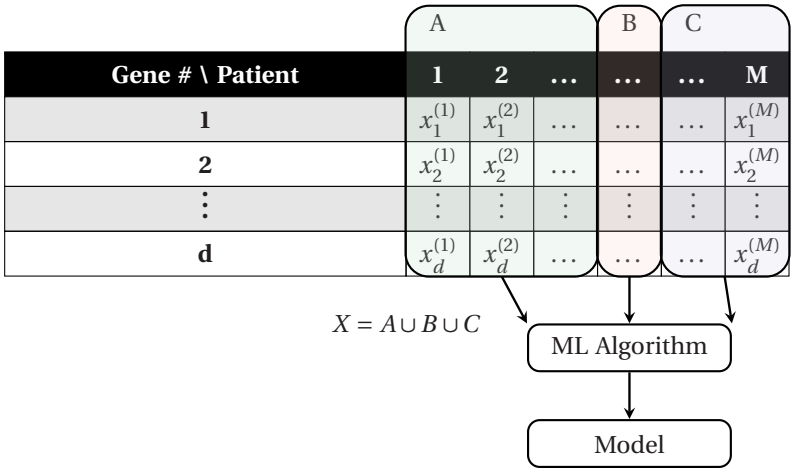

Figure 5.7: Collaborative machine learning (ML) model training. Rather than training an ML algorithm on one dataset such as part A, three dataset are first combined. Then ML is used. The data consists of $M = 472$ individuals, $d = 17,814$ genes for the BC-TCGA dataset, and $M = 225$, $d = 12,634$ for the GSE2034 dataset.

Given a number of patients M that each have a d-dimensional gene expression data vector and a label (healthy / not healthy), the goal is to construct a model that can correctly identify new patients as healthy or not based on their genome data. In the competition, two dataset were available: $M = 472$, $d = 17,814$ for the BC-TCGA dataset, and $M = 225$, $d = 12,634$ for the GSE2034 dataset. Many machine learning algorithms exist for this problem, and in general these benefit from having more data available. Therefore, if there are three parties A, B, C that all have a dataset for their own set of patients, it is in the interest of all parties to share their data, as this leads to more accurate models. See Fig. 5.7.

5.2.5. RANDOM FEATURES EXPANSION

Random feature expansions [55–57] (RFEs) are a type of machine learning model that are very easy and fast to train. Whereas other machine learning models such as neural networks or kernel methods would train all model weights or choose them according to the available data, in the RFE model most parameters are chosen randomly. Surprisingly, this approach still gives uniform approximation guarantees, meaning that any target function can be approximated arbitrarily well under general assumptions [58]. And because of the randomness in the model, the original data cannot be retrieved, as we show in Section 5.2.12. Similar random methods have been used as a linear dimensionality reduction technique on the dataset described in the previous subsection [44], without taking security into account.

Given data $\mathbf{x}_j \in \mathbb{R}^d, y_j \in \mathbb{R}$, $j = 1, \ldots, M$, an RFE is a nonlinear function $\text{RFE} : \mathbb{R}^d \to \mathbb{R}$ that approximates these data points after using a linear least squares type method. First, the model size D needs to be chosen, which can increase the accuracy of the model at the cost of more computation time. Then random parameters $\mathbf{W} \in \mathbb{R}^{D \times d}$ and $\mathbf{b} \in \mathbb{R}^D$, are

drawn from continuous probability distributions:

$$\begin{aligned} \mathbf{W}_{n,i} &\sim \text{Normal}(0,1), \\ \mathbf{b}_n &\sim \text{Uniform}[0,2\pi], \end{aligned} \tag{5.10}$$

for $n = 1,\ldots,D$, $i = 1\ldots,d$. Random features $\mathbf{z}_j \in \mathbb{R}^D$ are then constructed from the data as follows:

$$\mathbf{z}_j = \cos(\frac{1}{d}\mathbf{W}\mathbf{x}_j + \mathbf{b}), \tag{5.11}$$

for $j = 1,\ldots,M$. The cosine is taken element-wise. Now the learning algorithm is defined by solving the regularized linear least squares problem

$$\mathbf{c}^* = \arg\min_{\mathbf{c}} \sum_{j=1}^{M} ||\mathbf{z}_j \cdot \mathbf{c} - y_j||^2 + \lambda||\mathbf{c}||^2 \tag{5.12}$$

where $\lambda > 0$ is a regularization parameter which is needed to avoid overfitting, to deal with noise in the data, and for numerical stability. The minimization problem (5.12) has the following closed-form solution:

$$\begin{aligned} \mathbf{X} &= [\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(M)}], \\ \mathbf{Y} &= [y^{(1)},\ldots,y^{(M)}], \\ \mathbf{Z} &= \cos(\frac{1}{d}\mathbf{W}\mathbf{X} + \mathbf{b}), \\ \mathbf{c}^* &= (\mathbf{Z}\mathbf{Z}^T + \lambda I_{D\times D})^{-1}\mathbf{Z}\mathbf{Y}, \end{aligned} \tag{5.13}$$

with I the identity matrix. After training, the following equation can be used on a new patient with gene data $\mathbf{x}_{new}$ to let the model predict whether the new patient is healthy or not:

$$\text{RFE}(\mathbf{x}_{new}) = \cos(\frac{1}{d}\mathbf{W}\mathbf{x}_{new} + \mathbf{b}) \cdot \mathbf{c}^*. \tag{5.14}$$

5.2.6. SECURE MULTIPARTY COMPUTATION

A secure multi-party computation is an interactive cryptographic protocol that allows for two or more mistrusting parties to jointly compute a function using their privately contributed data as input [59–61]. Depending on the exact scenario, it may allow for the output of the desired function to be public, but the contributed inputs should be private with the assumption that each party does not digress from the rules of the protocol. Secure multi-party computations are designed in a semi-honest security model. This is a security model where all parties are required to follow the description of the protocol, without being able to act maliciously. However, there exist literature that describe the design of secure multi-party computation protocols which are secure under the malicious model [62, 63]. Some examples of secure multi-party computation protocols can be seen in [63–66]. Secure multiparty computation protocols are usually computationally cheaper when compared to other techniques such as homomorphic encryption, but they suffer from communication overheads due to the multiple interactions require to compute the functions.

5.2.7. NOTATION

In table A.1 we provide a list of notations and corresponding description.

Table 5.4: Notations

Notation	Description
$\mathcal{N}$	Number of parties in the protocol.
$party$	Each participants of the protocol, labelled $\{A, B, C\}$
M_{party}	Size of the record contributed by a party
M	Sum of all records from all parties $\sum M_{party}$
d	Number of gene loci considered for each protocol
$\mathbf{W}$	Matrix of random number with elements, $\leftarrow (0,1)$
$\mathbf{b}_{party}$	$D \times 1$ Matrix of random numbers, $\leftarrow [0, 2\pi]$
$\mathbf{b}$	Average of matrix from all parties $\left(\sum b_{party}/\right)\mathcal{N}$
$\mathbf{B}'(M_{party})$	$1 \times M_{party}$ matrix with each entry set to 1
$\mathbf{B}$	Security parameter, with default value of 80
$\mathbf{Y}_{party}$	Array of response variables for a party
$\mathbf{Y}$	Array of concatenated response variables from all N parties
$\mathbf{X}_{party}$	Feature matrix from a party
$\mathbf{X}$	Concatenation of all feature matrices from all N parties
$\mathbf{X}^t$	Transpose of the matrix $\mathbf{X}$
$Hash()$	A secure hash function such as SHA-256 $\mathbf{X}$
$min(\mathbf{X}), max(\mathbf{X})$	The minimum and maximum elements in the matrix $\mathbf{X}$
$regParam$	Regulation parameter for adjusting the trained model, $regParam \in [-1,1]$
$Iden(i)$	A square identity matrix with size i
$Inv(\mathbf{X})$	Inverse of the matrix $\mathbf{X}$

5.2.8. SCOTML

In this section, we introduce the problem setting and describe our proposed solution (SCOTML) and discuss its correctness and possible optimizations.

5.2.9. PROTOCOL SETTINGS AND ASSUMPTIONS

For every model to be generated, participants in the protocol are pre-decided. Each party is assumed to own a private set of dataset which constitutes patients of the target disease to serve as cases, and non-patients to serve as control set. The protocol as seen in Fig. 5.8 is set in an honest-but-curious security model, and all parties are assumed to follow the protocol description. Also, each party is required to have a threshold of patients in order to participate, this is to avoid the possibility of a party joining the protocol with only control dataset. For every setting of the protocol with N parties, there must be at least $\frac{N}{2}+1$ honest participants to guarantee the privacy of the private data and the model generated. The response values of each record, which represents the status of a record with respect to the disease of interest is encoded as $\{-1, 1\}$ representing healthy and sick respectively.

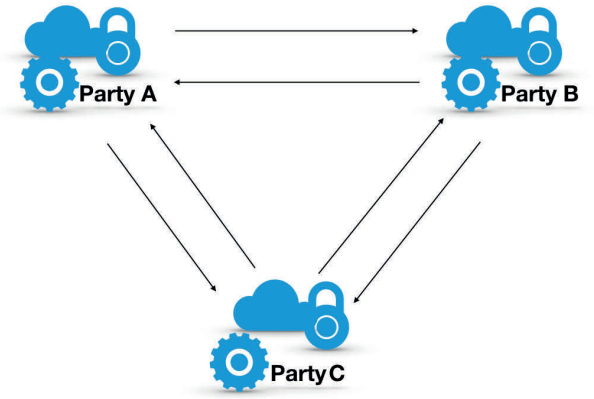

Figure 5.8: Protocol Setting

5.2.10. PROTOCOL DESCRIPTION

The steps of the SCOTML protocol as shown in Fig. 5.7 is described as follows:

Step 1. Initialization of parameters, where N is the number of participant for the protocol. Values for $(d, D, regParam)$ are jointly agreed upon by all parties, and all N parties are assigned unique identifiers as well as their position on the queue. This queuing is meant to allow for alignment of data by each party, when data is received from other $(N-1)$ sources.

Step 2. Each party determines the number of records (M_{party}) being the size of dataset which they wish to contribute to the protocol. Each party now has a local dataset of dimension $(d \times M_{party})$ which is private to them.

Step 3. Each party privately shuffles their private dataset of M_{party} records for every execution of the protocol, in order to mitigate against possible correlation of records across multiple runs of the protocol.

Step 4. Each party independently generates a matrix $\mathbf{W}$ of random numbers following (5.10), with dimension $D \times d$. Each value in $\mathbf{W}$ is reduced by d, $\mathbf{W} = \mathbf{W}/d$.

Step 5. Each party independently generates a matrix $\mathbf{b}_{party}$ of random numbers following (5.10), with dimension $D \times 1$.

Step 6. Each party privately generates a random noise matrix $(\hat{\mathbf{X}}_{party})$ of dimensional $(M_{party} \times d)$, with elements $\in [-0.083, 0.083]$. And sets the new value of the feature matrix to $\mathbf{X}_{party} = \mathbf{X}_{party} + \hat{\mathbf{X}}_{party}$.
The constant value 0.083 is chosen to be small but determined by distribution of the data to be trained.

Step 7. All $\mathcal{N}$ parties communicate a commitment of their random numbers by broadcasting $(M, Hash(\mathbf{W}_{party}), Hash(\mathbf{b}_{party}))$. This commitment prevents any possible changes to the values of the random numbers when transmitted in subsequent steps.

Step 8. All parties securely broadcast their values of $(\mathbf{b}_{party}, \mathbf{W}_{party})$.

Step 9. Upon receiving $(\mathbf{b}_{party}, \mathbf{W}_{party})$, from all participants and verifying against the commitments, each party node proceeds to compute

$$\mathbf{W} = \left(\sum^{\mathcal{N}} \mathbf{W}_{party}\right) / \mathcal{N} \tag{5.15}$$

$$\mathbf{b} = \left(\sum^{\mathcal{N}} b_{party}\right) / \mathcal{N} \tag{5.16}$$

$$\mathbf{B}_{party} = \mathbf{b} \times \mathbf{B}'(M_{party}) \tag{5.17}$$

$$\mathbf{H}_{party} = (\mathbf{W} \times \mathbf{X}^t_{party}) + \mathbf{B}_{party} \tag{5.18}$$

$$\mathbf{Z}_{party} = cosine(H_{party}) \tag{5.19}$$

Step 10. Values of $(\mathbf{Z}_{party}, \mathbf{Y}_{party})$ are broadcast to all parties.

Step 11. After receiving $(\mathbf{Z}_{party}, \mathbf{Y}_{party})$ from all parties, each party concatenates the matrices according to the agreed alignment in **Step 1.** to obtain $\mathbf{Z}$ and $\mathbf{Y}$.

Step 12. Each party proceeds with the following computation locally:

$$\mathbf{C}_0 = regParam \times Iden(M) \tag{5.20}$$

$$\mathbf{C}_1 = \mathbf{Z} \times \mathbf{Z}^t \tag{5.21}$$

$$\mathbf{C}_1 = \mathbf{C}_1 + \mathbf{C}_0 \tag{5.22}$$

$$\mathbf{C}_1 = Inv(\mathbf{C}_1) \tag{5.23}$$

$$\mathbf{C}_2 = \mathbf{Z} \times \mathbf{Y} \tag{5.24}$$

$$\mathbf{C} = \mathbf{C}_1 \times \mathbf{C}_2 \tag{5.25}$$

The set of values $(\mathbf{C}, \mathbf{W}, \mathbf{b})$ are then preserved as parameters of the model, which can be used in the prediction phase for classifying features from new patients.

Step 13. After the model generation phase, each party can privately compute the prediction phase without interacting with the multi-party computation protocol. This is done using the following equation: given a matrix of new features $\mathbf{X}_{new}$ of M_{new} records

$$\mathbf{Z}_{test} = cosine\left((\mathbf{W} \times \mathbf{X}^t_{new}) + (\mathbf{b} \times \mathbf{B}'(M_{new}))\right) \tag{5.26}$$

$$\hat{\mathbf{Y}}_{predicted} = \mathbf{Z}_{test} \times \mathbf{C} \tag{5.27}$$

A threshold is chosen with which to classify values in $\mathbf{Y}$.

Table 5.5: The computational complexity for training the model

Step	Hash.	Sc. Mul	Mat. Mul.	Mat. Inv.	Mat. Add.	Shuffle
3	–	–	–	–	–	M_{party}
4	–	$2(d*D)$	–	–	–	–
5	–	–	–	–	D	–
6	–	$M_{party}*d$	–	–	$M_{party}*d$	–
7	$D+(D*d)$	–	–	–	–	–
9	–	$D*d+D$	$D*M_{party}+D*d*M_{party}$	–	$2(D*d)+2D+2(D*M)$	–
12	–	M	$D*M*D+D*M+D*D$	D^3	$D*D$	–
Total	$D+Dd$	$3dD+M_{party}d$ $+M+D$	$DM_{party}(1+d)$ $+DM(D+1)+D^2$	D^3	$3D+M_{party}d$ $+2D(d+M)+D^2$	M_{party}

5.2.11. COMPLEXITY

The computational and communication complexity of our protocol is tabulated in Tab. 5.5. The complexity as shown depicts the cost to a single node/party in the protocol. From Tab. 5.5 the computational complexity can be expressed as $O(MD^2 + M_{party}Dd + D^3)$. This represents the number of operations to be computed in each node. Therefore, for $\mathcal{N}$ participants where each participant is assumed to contribute the same number of records, the computational complexity is multiplied $\mathcal{N}$ times. While the complexity depends on the size of the feature matrix, which in this work is the number of genes and the total records, the complexity is significantly determined by the number of random features D . While it is possible to reduced the value of D in order to gain computational advantage, it should be noted that D is inversely proportional to the accuracy of the trained model. Note that the term D^3 comes from a single matrix inversion operation, which can be avoided for example by making use of a recursive least squares approach [67].

There are 7 rounds of communication which occur during the course of the protocol execution, these are found in steps $(1,7,8,10)$. This low rounds of communication results in the $O(D+Dd+M_{party}+DM_{party})$ communication complexity where we are presented with a linear complexity in the size of D and d .

5.2.12. SECURITY AND PRIVACY

Security/Privacy proof for the protocol is argued under the semi-honest security setting and with all relevant assumptions clearly stated. All parties wish to hide the values of the genes relevant to the test, but not the gene loci, since all participants share the loci of interest. However, the response value $\mathbf{Y}$ is assumed to be known by other participants because they all broadcast their records of case and control set. The motive of a compromised participant is to derive information about that feature matrix ($\mathbf{X}$) of other parties.

Information about $\mathbf{X}$ values are transmitted or shared in the form of $\mathbf{Z}$, therefore our proof of protection of the privacy of the feature matrix is achieved as follows: Given the random feature matrix $\mathbf{Z}$ it should be computationally impossible for an adversary with polynomially bounded resources to retrieve the feature matrix $\mathbf{X}$. First we achieve a dimension reduction from d (17,814) for the BC-TGCA dataset to (D) (400). This reduction makes it more difficult to recover the original information that was lost. We provide the security and privacy argument using 3 main countermeasures.

1. Addition of noise in Step. 6 simulate Learning With Error problems (LWE). This countermeasure adds a bounded noise which is only known to the data owner,

thereby obfuscating the original private data.

2. Insufficient equations to solve for all variables, leads to information theoretic security argument. By reducing the dimension of the matrix, and keeping all relevant information about **X** private, it is impossible for an adversary to correctly solve for d variables using D equations since $d >> D$.

3. Scrambling property introduced using cosine function. While the $Cosine$ function does not independently obfuscate input data, it adds a layer of privacy protection to the protocol, making it even more difficult for an adversary to derive the values in **X** given only Z. This is evident owing to the non-bijective properties of the $Cosine$ function.

Table 5.6: Results compared to deep learning approach

	Dataset	Avg runs	Acc.	time (sec.)	P-P	SMC
Benchmark1	GSE2034	×100	69%	980	No	No
SCOTML	GSE2034	×1000	73%	39	Yes	Yes
Benchmark2	BC-TCGA	×100	95.63%	1303.9	No	No
SCOTML	BC-TCGA	×1000	99.48%	53	Yes	Yes

5.2.13. IMPLEMENTATION AND RESULTS

A description of the implementation techniques, libraries, and tools that were utilized to realize the prototype of SCOTML and the corresponding results are presented in this section. We present a SCOTML prototype that compares complexity and accuracy of SCOTML against existing non privacy-preserving techniques using the same dataset. Given the data presented in Section 5.2.4, we are also provided with two benchmark (Benchmark1, Benchmark2) implementations for both dataset. Benchmark1 and Benchmark2 are both deep learning implementations for model generation and prediction using the GSE2034 and BC-TCGA dataset respectively. Our prototype implementation of SCOTML is achieved using Python3 with Numpy, pyftpdlib, and ftplib modules. The distributed implementation is achieved using docker containers which allows for easy setup on any commodity hardware. The results, both in accuracy and time complexity are shown in Table 5.6. It can be seen that SCOTML achieves higher accuracy than both benchmarks in only a fraction of the time. Note that while we compare SCOTML implementations to the deep learning implementations of Benchmark1 and Benchmark2, the benchmark implementations do not provide for privacy-preserving features, neither do they allow for a secure multiparty computation. In the benchmark implementation, it is assumed that there is a trusted party to whom all parties contribute their private data to, and this trusted party is responsible for training the model and returning results to all parties. As such, SCOTML has outperformed the benchmark methods on three fronts, namely accuracy, computational efficiency, and privacy, especially since the benchmark had no privacy property whatsoever.

5.2.14. CONCLUSION

SCOTML is a privacy-preserving machine learning solution that aids collaborative machine learning in an era of cloud based computation and storage services. Our prototype is implemented to allow 3 or more mistrusting parties to collaborate in generating public ML models while provably preserving the privacy of their contributed genome data. This setting and solution can aid multiple bodies to contribute sensitive genome data for the purpose of research without having to worry about privacy constraints. Our protocol is scale-able for our test dataset of 472 records each having 17,814 genes, it takes less than 55 seconds to complete training on a commodity hardware while being more accurate, faster, and more secure than the existing deep learning approach.

REFERENCES

[1] K. Sikorska, E. Lesaffre, P. F. Groenen, and P. H. Eilers, *GWAS on your notebook: fast semi-parallel linear and logistic regression for genome-wide association studies,* BMC bioinformatics **14**, 166 (2013).

[2] K. Michailidou, S. Lindström, J. Dennis, J. Beesley, S. Hui, S. Kar, A. Lemaçon, P. Soucy, D. Glubb, A. Rostamianfar, *et al., Association analysis identifies 65 new breast cancer risk loci,* Nature **551**, 92 (2017).

[3] M. Capasso, M. Devoto, C. Hou, S. Asgharzadeh, J. T. Glessner, E. F. Attiyeh, Y. P. Mosse, C. Kim, S. J. Diskin, K. A. Cole, *et al., Common variations in bard1 influence susceptibility to high-risk neuroblastoma,* Nature genetics **41**, 718 (2009).

[4] T. A. Pearson and T. A. Manolio, *How to interpret a genome-wide association study,* Jama **299**, 1335 (2008).

[5] P. M. Visscher, M. A. Brown, M. I. McCarthy, and J. Yang, *Five years of GWAS discovery,* The American Journal of Human Genetics **90**, 7 (2012).

[6] 1000 Genomes Project Consortium, *A global reference for human genetic variation,* Nature **526**, 68 (2015).

[7] NLM, *Single nucleotide polymorphism,* (2018), `https://ghr.nlm.nih.gov/primer/genomicresearch/snp` Online; accessed January, 2018.

[8] S. Kathiresan, O. Melander, D. Anevski, C. Guiducci, N. P. Burtt, C. Roos, J. N. Hirschhorn, G. Berglund, B. Hedblad, L. Groop, *et al., Polymorphisms associated with cholesterol and risk of cardiovascular events,* New England Journal of Medicine **358**, 1240 (2008).

[9] X. Wan, C. Yang, Q. Yang, H. Xue, N. L. Tang, and W. Yu, *Megasnphunter: a learning approach to detect disease predisposition snps and high level interactions in genome wide association study,* BMC bioinformatics **10**, 1 (2009).

[10] M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang, *Secure logistic regression based on homomorphic encryption: Design and evaluation,* JMIR medical informatics **6** (2018).

[11] M. Kim and K. Lauter, *Private genome analysis through homomorphic encryption,* in *BMC medical informatics and decision making*, Vol. 15 (BioMed Central, 2015) p. S3.

[12] C. Ugwuoke, Z. Erkin, and R. L. Lagendijk, *Privacy-safe linkage analysis with homomorphic encryption,* in *Signal Processing Conference (EUSIPCO), 2017 25th European* (IEEE, 2017) pp. 961–965.

[13] K. Lauter, A. López-Alt, and M. Naehrig, *Private computation on encrypted genomic data,* in *Progress in Cryptology-LATINCRYPT 2014* (Springer, 2014) pp. 3–27.

[14] D. G. Kleinbaum and M. Klein, *Logistic regression: a self-learning text* (Springer Science & Business Media, 2010).

[15] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, *A survey of machine learning for big data processing,* EURASIP Journal on Advances in Signal Processing **2016**, 67 (2016).

[16] C. Bonte and F. Vercauteren, *Privacy-preserving logistic regression training,* BMC medical genomics **11**, 86 (2018).

[17] K. Estrada, A. Abuseiris, F. G. Grosveld, A. G. Uitterlinden, T. A. Knoch, and F. Rivadeneira, *Grimp: a web-and grid-based tool for high-speed analysis of large-scale genome-wide association using imputed data,* Bioinformatics **25**, 2750 (2009).

[18] S. S. Shringarpure and C. D. Bustamante, *Privacy risks from genomic data-sharing beacons,* The American Journal of Human Genetics **97**, 631 (2015).

[19] 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* (ACM, 2013) pp. 95–106.

[20] G. Danezis and E. De Cristofaro, *Fast and private genomic testing for disease susceptibility,* in *Proceedings of the 13th Workshop on Privacy in the Electronic Society* (ACM, 2014) pp. 31–34.

[21] J. H. Cheon, A. Kim, M. Kim, and Y. Song, *Homomorphic encryption for arithmetic of approximate numbers,* in *International Conference on the Theory and Application of Cryptology and Information Security* (Springer, 2017) pp. 409–437.

[22] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, *Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy,* in *International Conference on Machine Learning* (2016) pp. 201–210.

[23] L. Buzdugan, M. Kalisch, A. Navarro, D. Schunk, E. Fehr, and P. Bühlmann, *Assessing statistical significance in multivariable genome wide association analysis,* Bioinformatics **32**, 1990 (2016).

[24] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein, *Logistic regression* (Springer, 2002).

[25] A. Agresti and M. Kateri, *Categorical data analysis* (Springer, 2011).

[26] P. Paillier, *Public-key cryptosystems based on composite degree residuosity classes,* in *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding* (1999) pp. 223–238.

[27] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, *(leveled) fully homomorphic encryption without bootstrapping,* ACM Transactions on Computation Theory (TOCT) **6**, 1 (2014).

[28] Z. John Lu, *The elements of statistical learning: data mining, inference, and prediction,* Journal of the Royal Statistical Society: Series A (Statistics in Society) **173**, 693 (2010).

[29] E. C. Hayden, *Is the $1,000 genome for real?* Nature News (2014).

[30] J. Li, T. Zhao, Y. Zhang, K. Zhang, L. Shi, Y. Chen, X. Wang, and Z. Sun, *Performance evaluation of pathogenicity-computation methods for missense variants,* Nucleic acids research **46**, 7793 (2018).

[31] C. T. Johansen, J. Wang, M. B. Lanktree, H. Cao, A. D. McIntyre, M. R. Ban, R. A. Martins, B. A. Kennedy, R. G. Hassell, M. E. Visser, *et al.*, *Excess of rare variants in genes identified by genome-wide association study of hypertriglyceridemia,* Nature genetics **42**, 684 (2010).

[32] Y. Shi, H. Zhao, Y. Shi, Y. Cao, D. Yang, Z. Li, B. Zhang, X. Liang, T. Li, J. Chen, *et al.*, *Genome-wide association study identifies eight new risk loci for polycystic ovary syndrome,* Nature genetics **44**, 1020 (2012).

[33] J. Zou, M. Huss, A. Abid, P. Mohammadi, A. Torkamani, and A. Telenti, *A primer on deep learning in genomics,* Nature genetics **51**, 12 (2019).

[34] S. Uppu, A. Krishna, and R. P. Gopalan, *A deep learning approach to detect snp interactions.* JSW **11**, 965 (2016).

[35] J. O. Ogutu, T. Schulz-Streeck, and H.-P. Piepho, *Genomic selection using regularized linear regression models: ridge regression, lasso, elastic net and their extensions,* in *BMC proceedings*, Vol. 6 (BioMed Central, 2012) p. S10.

[36] F. Han and W. Pan, *Powerful multi-marker association tests: unifying genomic distance-based regression and logistic regression,* Genetic epidemiology **34**, 680 (2010).

[37] S. Shen, S. Tople, and P. Saxena, *A uror: defending against poisoning attacks in collaborative deep learning systems,* in *Proceedings of the 32nd Annual Conference on Computer Security Applications* (ACM, 2016) pp. 508–519.

[38] K. Xu, H. Ding, L. Guo, and Y. Fang, *A secure collaborative machine learning framework based on data locality,* in *2015 IEEE Global Communications Conference (GLOBECOM)* (IEEE, 2015) pp. 1–5.

[39] M. Imani, Y. Kim, S. Riazi, J. Messerly, P. Liu, F. Koushanfar, and T. Rosing, *A framework for collaborative learning in secure high-dimensional space,* in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)* (IEEE, 2019) pp. 435–446.

[40] J. Yuan and S. Yu, *Privacy preserving back-propagation neural network learning made practical with cloud computing,* IEEE Transactions on Parallel and Distributed Systems **25**, 212 (2013).

[41] L. Ohno-Machado, V. Bafna, A. A. Boxwala, B. E. Chapman, W. W. Chapman, K. Chaudhuri, M. E. Day, C. Farcas, N. D. Heintzman, X. Jiang, *et al.*, *idash: integrating data for analysis, anonymization, and sharing,* Journal of the American Medical Informatics Association **19**, 196 (2011).

[42] P. Larranaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armananzas, G. Santafé, A. Pérez, *et al.*, *Machine learning in bioinformatics,* Briefings in bioinformatics **7**, 86 (2006).

[43] S. Mitra, S. Datta, T. Perkins, and G. Michailidis, *Introduction to machine learning and bioinformatics* (Chapman and Hall/CRC, 2008).

[44] H. Xie, J. Li, Q. Zhang, and Y. Wang, *Comparison among dimensionality reduction techniques based on random projection for cancer classification,* Computational biology and chemistry **65**, 165 (2016).

[45] E. Frank, M. Hall, L. Trigg, G. Holmes, and I. H. Witten, *Data mining in bioinformatics using weka,* Bioinformatics **20**, 2479 (2004).

[46] P. Mohassel and Y. Zhang, *Secureml: A system for scalable privacy-preserving machine learning,* in *2017 IEEE Symposium on Security and Privacy (SP)* (IEEE, 2017) pp. 19–38.

[47] K. Chaudhuri and C. Monteleoni, *Privacy-preserving logistic regression,* in *Advances in neural information processing systems* (2009) pp. 289–296.

[48] R. Shokri and V. Shmatikov, *Privacy-preserving deep learning,* in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security* (2015) pp. 1310–1321.

[49] A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon, *Logistic regression model training based on the approximate homomorphic encryption,* BMC medical genomics **11**, 83 (2018).

[50] J. H. Cheon, D. Kim, Y. Kim, and Y. Song, *Ensemble method for privacy-preserving logistic regression based on homomorphic encryption,* IEEE Access **6**, 46938 (2018).

[51] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, *Chameleon: A hybrid secure computation framework for machine learning applications,* in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security* (2018) pp. 707–721.

[52] L. Van De Kamp, C. Ugwuoke, and Z. Erkin, *Economy: Ensemble collaborative learning using masking,* in *2019 IEEE 30th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC Workshops)* (IEEE, 2019) pp. 1–6.

[53] T. G. Dietterich, *Ensemble methods in machine learning,* in *International workshop on multiple classifier systems* (Springer, 2000) pp. 1–15.

[54] H. Tang and X. Wang, *Track iv: Secure collaborative training of machine learning model,* (2019), `http://www.humangenomeprivacy.org/2019/competition-tasks.html` Online; accessed October, 2019.

[55] A. Rahimi and B. Recht, *Random features for large-scale kernel machines,* in *Advances in neural information processing systems* (2008) pp. 1177–1184.

[56] A. Rahimi and B. Recht, *Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning,* in *Advances in neural information processing systems* (2009) pp. 1313–1320.

[57] L. Bliek, H. R. Verstraete, M. Verhaegen, and S. Wahls, *Online optimization with costly and noisy measurements using random Fourier expansions,* IEEE transactions on neural networks and learning systems **29**, 167 (2016).

[58] A. Rahimi and B. Recht, *Uniform approximation of functions with random bases,* in *2008 46th Annual Allerton Conference on Communication, Control, and Computing* (IEEE, 2008) pp. 555–561.

[59] C. Hazay and Y. Lindell, *Efficient secure two-party protocols: Techniques and constructions* (Springer Science & Business Media, 2010).

[60] O. Goldreich, *Secure multi-party computation,* Manuscript. Preliminary version , 86 (1998).

[61] R. Cramer, I. B. Damgård, *et al.*, *Secure multiparty computation* (Cambridge University Press, 2015).

[62] Y. Lindell and B. Pinkas, *An efficient protocol for secure two-party computation in the presence of malicious adversaries,* in *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, 2007) pp. 52–78.

[63] P. Mohassel and M. Franklin, *Efficiency tradeoffs for malicious two-party computation,* in *International Workshop on Public Key Cryptography* (Springer, 2006) pp. 458–473.

[64] C. Ugwuoke, Z. Erkin, and R. L. Lagendijk, *Secure fixed-point division for homomorphically encrypted operands,* in *Proceedings of the 13th International Conference on Availability, Reliability and Security* (2018) pp. 1–10.

[65] T. Schneider and M. Zohner, *GMW vs. Yao? efficient secure two-party computation with low depth circuits,* in *International Conference on Financial Cryptography and Data Security* (Springer, 2013) pp. 275–292.

[66] Y. Huang, D. Evans, J. Katz, and L. Malka, *Faster secure two-party computation using garbled circuits.* in *USENIX Security Symposium*, Vol. 201 (2011) pp. 331–335.

[67] A. H. Sayed and T. Kailath, *Recursive least-squares adaptive filters,* The Digital Signal Processing Handbook **21** (1998).

6

Discussion and Conclusion

In this concluding chapter we present our evaluations and concluding thoughts on the various cryptographic privacy-preserving proposals to the identified privacy threats inherent with the processing of genome data.

6.1. DISCUSSION

In a society where genome data processing is gaining traction due to it various application areas, the privacy of the genome data is a relevant conversation to have. More so, with laws and regulations such as HIPAA and GDPR that regulate use of health data, the need for provably secure measures for storing and processing sensitive and health related data becomes paramount in a COVID-19 and a post-COVID-19 society.

In this thesis, we explored the wide range of cryptographic solutions that are applicable to various scenarios that covers storage, processing and publishing of digitized genome data. Having completed the journey from a formulated problem statements through evaluation of proposed solutions, it is important to reflect on the path we have taken, why and how the choices were made.

In Chapter 3, we first present a generic survey that maps out various scenarios that present identified privacy and security challenges that have been addressed in literature, covering both research and industry works. This equally attests to the importance of investigating privacy measures for processing the genome and in extension validates the problem discussed in the thesis. In the next paper we present a demonstration of how privacy enhancing technologies listed in Chapter 2 can be utilized in addressing the privacy requirements in computing linkage analysis, which is a genome data processing technique. In the last paper of Chapter 3 we present a general discussion on genome wide association studies and some of the algorithms that are realizable in a privacy-preserving mode.

While Chapter 3 focuses on algorithm and operations that are peculiar to the genome wide association study phase, Chapter 4 explores the possibility of encapsulating post-GWAS algorithms with privacy-preserving techniques. We do this by showing that the results obtained from a GWAS could be further utilized to provide services such as disease susceptibility testing as a service. This implementation is secure and light weight, making it a prospect for industrial application already.

In Chapter 5 we explore the use of machine learning algorithms in realizing the services of processing the genome dataset. However, because the base or classical machine learning algorithms are not usually designed with privacy and security as a core feature, we are posed with the challenge of redesigning a privacy-preserving variants of select machine learning algorithms that are used for genome data processing. We go ahead to show that not only is the realization of privacy-preserving algorithms possible, but that they could in some cases outperform some non-privacy-preserving variants in accuracy, performance and complexity. We also present the possibility of collaboratively computing machine learning models for more than two mistrusting parties.

In the works presented in this thesis we show a variety of privacy enhancing techniques by applying them differently as the scenarios require. We have also shown that while in some cases a single technique would out-rightly be sufficient, refer to Chapter 4, in some other cases, a hybrid or cocktail of techniques become required for attaining the privacy and utility goal of the problem statement.

6.2. EVALUATION

Having provided a problem statement and the relevance of provable privacy-preserving genome data processing which reads:

For every party in our defined setting/scenario of genome data processing, how can we utilize cryptographic primitives to provide provable privacy and security for all identified assets within the threat model, and still realize the utility of the protocol as a relatively efficient alternative?

We present our opinion on the state-of-the-art in both the research community and the industry. We list our intended objectives and try to match them with solution where we have realised those objectives.

1. **Privacy and Security:** How do we design protocols with the aim of providing provable privacy and security guarantees, and equally optimize the computational, communication, and storage cost of realizing such a protocol? The privacy and security objective was realized in all attempted scenarios which can be found in Chapters 3, 4, 5.

2. **Acceptable Accuracy:** How can we preserve the utility of genome data services even in the protected domain, such that our privacy-preserving variants can replicate the accuracy obtainable in the non-protected variants of the protocols. Although the accuracy of a privacy-preserving model is commonly expected to perform worse than the plaintext solution, we show in the following works (REDACT: Chapter 5, SCOTML: Chapter 5 PREDICT: Chapter 4) that with the right choice of algorithm and design, a privacy-preserving version must not always perform worse than the non-privacy-preserving counterparts.

3. **Performance:** How can we realize privacy-preserving protocols that are not bedeviled by poor performances, but rather enhance performance by supporting efficient storage, acceptable communication complexity, and practical computational efficiency. In the following works (SCOTML: Chapter 5 , PREDICT: Chapter 4), we provide solutions that rival the performance achieved in the non-privacy-preserving solutions. This further validates our claim that some privacy-preserving solutions to genome data processing are already suitable for the wild due to their ease of scalability.

6.3. OPEN PROBLEMS

Notwithstanding that the state-of-the-art solutions have gone to great lengths to address the privacy concerns, there are still a trove of open problems that require contributions from science in order to find solutions. Some of those include:

1. Plug the loophole that leaves the privacy of the genome dataset to the discretion of the sequencer. Presently, this phase of the genome data life cycle requires the

use of biological samples and is left to the expert who has no cryptographic provable measures to constrain them from abusing the data. This remains an open problem.

2. Explore the possibility of significantly reducing the overhead incurred while computing privacy-preserving GWAS/machine learning in a non-interactive setting. Computing freshly protected genome data would often adopt the homomorphic encryption technique. However, this technique is expensive in both computation complexity and storage complexity due to inherent data expansion. This therefore makes them sometimes not suitable for scaling.

3. Enhancing the prospects of privacy-preserving collaboration between genome data owners without having to require active participation during the process of executing the algorithm. Most collaborative solutions are designed in the interactive model. This means that during some process of the computation, data would have to be exchanged between the computing party and some other entity. Which means that in the event of a loss in communication, the process cannot be completed. Not to mention the communication overhead that is associated with this. A compact non-interactive solution remains an open problem.

6.4. CONCLUSION

In the thesis, we have shown how genome data processing can be achieved within an acceptable privacy and utility criteria. We have presented the life cycle of a typical genome dataset and the algorithms that are commonly used in this process, and have shown that where no off-the-shelf solution exists, a cryptographic provable privacy-preserving solution can be designed. We demonstrate this with a proof of concept, designs and implementation as our contribution to to the field of privacy-preserving genome data processing. Finally we mention that the relevance and novelty of some of our solution can easily be measured by its consistent performance at the yearly iDASH challenge, where we presents our solutions to compete amongst other solutions from research institutions and companies.

A

APPENDIX SECURE FIXED-POINT DIVISION FOR HOMOMORPHICALLY ENCRYPTED OPERANDS

Due to privacy threats associated with computation of outsourced data, processing data on the encrypted domain has become a viable alternative. Secure computation of encrypted data is relevant for analysing dataset in areas (such as genome processing, private data aggregation, cloud computations) that require basic arithmetic operations. Performing division operation over-all encrypted inputs has not been achieved using homomorphic schemes in non-interactive modes. In interactive protocols, the cost of obtaining an encrypted quotient (from encrypted values) is computationally expensive. To the best of our knowledge, existing homomorphic solutions on encrypted division are often relaxed to consider public or private divisor. We acknowledge that there are other techniques such as secret sharing and garbled circuits adopted to compute secure division, but we are interested in homomorphic solutions. We propose an efficient and interactive two-party protocol that computes the fixed-point quotient of two encrypted inputs, using an efficient and secure comparison protocol as a sub-protocol. Our proposal provides a computational advantage, with a linear complexity in the digit precision of the quotient. We provide proof of security in the universally composable framework and complexity analyses. We present experimental results for two cryptosystem implementations in order to compare performance. An efficient prototype of our protocol is implemented using additive homomorphic scheme (Paillier), whereas a non-efficient fully-homomorphic scheme (BGV) version is equally presented as a proof of concept and analyses of our proposal.

Parts of this chapter have been published as:
(1) Ugwuoke, C., Erkin, Z., & Lagendijk, R. L. (2018, August). Secure fixed-point division for homomorphi-

A

A.1. INTRODUCTION

In an era where outsourcing computation has been well adopted and widely implemented [1], concerns over the privacy of sensitive data outsourced to processing entities become the new paradigm [2, 3]. The choice of outsourcing data to processing entities has become common and rightly so, partly because of the enormous processing and storage resources often possessed by such infrastructures such as cloud services. These entities continue to process outsourced data even in the face of privacy concerns and sensitivity of the data being processed. It is easier and convenient for a data owner or client to store their data on a central infrastructure where they can easily access their data using multiple devices. In some scenarios the algorithms required for computing data are proprietary and are considered trade secrets that should not be entrusted with every client [4]. The processing entity does not have to be trusted with sensitive data of the data owners, even as data owners wish to utilise the services of the cloud without completely trading their privacy thereof [5]. It is common to observe scenarios that require interaction between a client data and an untrusted processing entity in various areas. Such areas include financial sector, secret balloting [6], genome data computation [7, 8], private data aggregation setting [9], private healthcare data analyses [10]. Researchers in the field of cryptography continue to address challenges involving private analyses of data with the use of various cryptographic techniques. Such techniques include secret sharing [11], garbled circuit [12], and encrypted computation [13]. Any of the techniques may be adopted depending on the dataset involved and the adversarial model.

For researchers to unambiguously process and analyse datasets in plaintext, basic arithmetic operations which include: addition, subtraction, multiplication and division are usually necessary in various steps of these algorithms. While it is computationally feasible and relatively cheap to replicate basic arithmetic operations on private data with the use of primitives such as secret sharing and garbled circuit, processing encrypted (using public key) data presents a different challenge altogether. For instance, if a cloud infrastructure is required to homomorphically compute an operation over an outsourced data which is sensitive. It is obvious that the intermediate data and the output data would be ciphertexts of the homomorphic scheme. Homomorphic encryption schemes are currently used in non-interactive settings to perform addition, subtraction and multiplication (in some cases), but not division. In order to obtain division, interactive settings are often adopted. Computations over encrypted data using HE schemes provide provable semantic security, making them good alternatives for providing privacy of outsourced data. We narrow our discussion to consider only data outsourced in encrypted format and processed in the encrypted domain. For this, we consider homomorphic encryption [13–15] as the cryptographic primitive of choice.

Our focus is on a setting where a cloud infrastructure stores dataset of encrypted integers. The cloud is required to compute an algorithm that requires the division of two encrypted inputs
$(Enc_{pk}(a), Enc_{pk}(b))$, which are the numerator and denominator respectively. The result (quotient) should be of the expression $Enc_{pk}(\frac{a}{b})$. There is also a second party who

cally encrypted operands. *In Proceedings of the 13th International Conference on Availability, Reliability and Security (pp. 1-10).*

is honest-but-curious, and holds the decryption key with which to assist the cloud in executing the protocol. We assume in our setting that the quotient of any such division should be a decimal number with known properties such as precision, interval and sign. Existing attempts to solve the encrypted division are commonly presented in various settings in literature [16–18] , and consequently produce different forms of quotient. It can be observed in the work by Veugen [16] , that the requirement for operands is relaxed to allow an encrypted input be divided with a public divisor. It then requires establishing a two-party protocol that produces the quotient as a final result. Veugen further presents a different approach which relaxes the operands requirement to have an encrypted input with a private divisor [19]. Other works consider the integer parts of the division, and scaling and rounding technique is introduced where precision of the result is of interest. A non-interactive solution that efficiently outputs the encrypted quotient of two encrypted operands is still an open problem. An efficient and scalable solution that involves all encrypted input with an encrypted quotient will find use in various fields of applied cryptography, for instance when the aggregation of encrypted inputs is of interest, or even for computing encrypted percentages, p-values, and other high precision decimal results.

Secret sharing technique does not exactly require encrypted inputs, and relies on multiple parties to keep shares of the secret. The scenario we choose to address does not fit into the requirement of secret sharing, at least not without further modifications. Garbled Circuit would also provide a computationally cheaper option than HE, but we note that the setting of our problem (including homomorphically encrypted data with public key) does not make GC suitable. We are interested in reducing the number of parties that are needed to interact, store and process the data. Including a circuit (algorithm) that should be reusable for multiple instances. So, we are proposing a two-party protocol that reduces the number of rounds that is needed for computing the division of encrypted integers. Computing on encrypted data offers computational security, fewer semi-trusted players which reduces the chances of having colluding players and in extension can easily provide privacy for the data. Performing division of encrypted data as operands in a non-interactive mode is still a non-trivial task. The following research works [16–18] choose to address the encrypted division problem in an interactive setting instead. Our solution is described in a semi-honest security setting. The setting is defined such that one party $Alice$ holds the encrypted operands $Enc_{pk}(a)$, $Enc_{pk}(b)$ and wishes to perform the division, while the other party Bob is in possession of the decryption key, and helps with decryption of ciphertexts at intermediary steps of the two-party protocol.

Contribution: In this work, we present an efficient and provably secure two-party protocol that takes as input two encrypted operands and outputs the encrypted quotient up to a desired decimal precision. Our motivation is drawn from the need to obtain a secure division protocol which allows for fixed-point results with high efficiency. We leverage on a state-of-the-art comparison protocol [20] that is secure, efficient and cryptosystem independent. With that, we construct a secure fixed-point integer division protocol which in extension is cryptosystem independent. Our construction guarantees privacy of the input operands and allows for fixed-point results. We compute the result by executing our protocol ρ rounds, with ρ being the digit precision of the quo-

tient. Each execution produces an encryption of a single digit $Enc_{pk}(q_i)$, corresponding to the same position on the ρ-digit precision quotient for $0 \leq i < \rho$. $Enc_{pk}(q_0)$ is the integer component of the fixed-point quotient, while $Enc_{pk}(q_1)\ldots Enc_{pk}(q_{\rho-1})$ represent the encryption of the decimal component up to the predefined digit precision ρ. Two implementations of our proposed construction are provided. First, is a Paillier [14] implementation, which is efficient and easily deployable due to its practicability. Second, is a somewhat homomorphic scheme-BGV [21] implementation, which is only provided as a proof of concept and for analysis of performance. We provide detailed security proof using the simplified universally composable security as introduced by Canetti, Cohen and Lindell [22]. Complexity analyses and implementation results that demonstrate the feasibility and efficiency of our proposed protocol.

A.2. RELATED WORKS

Here we provide a much detailed exposition of different works that have provided solutions to encrypted integer division, and how their contributions differ than our approach.

Bunn and Ostrovsky [23] adopt a security model for an honest-but-curious adversary, and propose a construction for privately computing k-mean clustering protocol, with the use of a two-party division. In their work, the operands are sampled from $\mathbb{Z}_N$. Given two inputs $P, D \in \mathbb{Z}_N$, the Division Algorithm produces the following : $Q < N$ and $0 \leq R < D$ such that $P = QD + R$ and (Q, R) are unique ordered pair in $\mathbb{Z}_N$. It becomes immediately evident that the division does not attempt to cover arbitrary input space, but inherits the upper bound of a finite message space in a semantically secure homomorphic encryption scheme. Their construction continues with defining Q as the quotient of the division (possibly rounded down to the nearest integer), while discarding R. In the implementation, both parties are provided with shares of the inputs such that $P = P^A + P^B$ and $D = D^A + D^B$ where A and B represents both parties. At completion, the division protocol is expected to produce a share of Q to both parties. Subsequently, other sub-protocols are run to securely compute the final value Q. It is important to note that although this construction successfully computes the k-mean, it does not provide for a floating point result.

Using the Paillier Homomorphic Encryption Scheme, Dahl et al. [17] demonstrate how to compute secure division for integer inputs. Unlike some constructions that assume private or public input for the divisor, a ciphertext is considered as the divisor here. Two different protocols are presented, whereby one provides for a constant-round of communication, the other offers a sub-linear communication complexity. Several subprotocols are relied on to achieve the division, which includes: *Prefix-or of a sequence of bits, bit decomposition of encrypted value, computing the greater-than relation.* Also, the authors demonstrate a division protocol that provides for an integer quotient, which does not consider floating point results.

Another interesting approach to securely computing integer division is elucidated by Veugen [16, 19]. The author introduces two possible constructions that result to different results. One approach computes the exact division result, after taking in as input an encrypted numerator, and a public divisor. The second approach computes an approximate division result, with inputs of an encrypted numerator with a choice of private

or public divisor. It is observed that the approximate result approach performs better than the exact result construction. Just like the previous work [23]. Veugen's construction leverages a secure comparison protocol to compute exact or approximate values for the quotients of two integer operands. However, the floating point quotient is also not addressed in his work.

Catrina and Saxena in [24] address challenges faced with outsourced computation by providing a multiparty protocol which securely computes rational numbers in a privacy-preserving manner. In a semi-honest model, their construction computes rational numbers represented in fixed-point, and deploys secret sharing as the cryptographic primitive. Their protocol provides for performing basic arithmetic operations which include addition, subtraction, multiplication and division. However, unlike the scenario considered in our proposal, Catrina and Saxena do not consider that the inputs are encrypted, which reduces the problem to computing division of private operands, hence the use of secret sharing. The different parties that keep shares of the secret only hold private data and not encrypted data, also the adopted technique induces extra rounds and communication costs for the entire protocol.

A similar work done by Franz et al. [18] adopts a two-party protocol for the computation of encoded real numbers in an oblivious way. The construction utilises a hybrid of homomorphic encryption using Paillier cryptographic scheme [14] and garbled circuit [12]. In their work, the resulting protocol is able to evaluate arithmetic operations including addition, subtraction, multiplication and division using a special type of encoding for real values representation. Franz et al. describe their work in a semi-honest setting, and the use of encrypted input provides computational security to the data being evaluated. However, the description of their work reveals that a precomputed lookup table is required in the protocol, which places certain constraints on the security and usability of their protocol. For instance, if the range of input values grows, it translate to growth in the lookup table which constitutes both storage and search cost for the parties. And when the input range becomes small, the probability of finding a collision in the lookup table increases when the same values are evaluated more than once, thereby degrading the security of the protocol.

Arguing that the continuous demand for outsourced computation is a motivation for secure computation, Aliasgari et al. [25] propose and demonstrate a solution for complex operations such as square root, logarithm, exponentiation and secure division, using private integer and non-integer values. They adopt secret sharing technique to privately evaluate the functions of interest, in a multi-party setting. However, it is important to mention then that the scope of their solution does not extend to encrypted data.

A.3. PRELIMINARIES

Here, we introduce the settings and cryptographic building blocks utilised for our construction. For all the two-party setting in this paper, we have two players $Alice$ and Bob, they run a secure two-party protocol whereby $Alice$ holds the public key and the encrypted inputs, while Bob holds the public and private key pair. Bob securely interacts with Alice to compute the result of the protocol over the encrypted domain.

A

A.3.1. HOMOMORPHIC ENCRYPTION

Homomorhpic Encryption (HE) allows for an arbitrary operation to be performed on ciphertexts, such that the resulting ciphertext would decrypt to the same value as would be obtained if a targeted algebraic operation were to be performed on the plaintext values. Let $Enc_{pk}(\cdot)$ and $Dec_{sk}(\cdot)$ represent encryption and decryption functions respectively. (m_1, m_2) are two messages and k is a scalar value, while $\boxplus$, $\boxdot$ and $\boxtimes$ are arbitrary operations on the ciphertexts. Then, homomorphism is defined as follows:

$$Dec_{sk}(Enc_{pk}(m_1) \boxplus Enc_{pk}(m_2)) = m_1 + m_2\,, \tag{A.1}$$

$$Dec_{sk}(Enc_{pk}(m_1) \boxdot Enc_{pk}(m_2)) = m_1 \cdot m_2\,, \tag{A.2}$$

$$Dec_{sk}(Enc_{pk}(m_1) \boxtimes k) = m_1 \cdot k\,. \tag{A.3}$$

PAILLIER SCHEME [14] :

Paillier cryptosystem is an additively homomorphic scheme which is secure under the computational composite residuosity assumption. Given a public key, private key pair (pk, sk) respectively. $pk := (g, n)$ and $sk := \lambda(n)$. Where $\lambda(n)$ is the Carmichael's function on n, define $\lambda(n) := lcm(p-1, q-1)$.

Encryption: $c := Enc_{pk}(m, r) := g^m \cdot r^n \ mod\ n^2$, where $c \in \mathbb{Z}^*_{n^2}$; $n := p \cdot q$, s.t p and q are distinct large primes, $r \leftarrow \mathbb{Z}^*_n$, g is generator of order n.

Decryption: Given ciphertext c, $m := \frac{\mathbb{L}_n(c^\lambda \ mod\ n^2)}{\mathbb{L}_n(g^\lambda \ mod\ n^2)} \ mod\ n$ and $\mathbb{L}_n(a) := \frac{a-1}{n}$.

Additive Homomorphism: Given two ciphertexts of messages $m_0\ and\ m_1$, we can compute the sum as follows:

$$\begin{aligned} Enc_{pk}(m_0, r_0) \times Enc_{pk}(m_1, r_1) &:= (g^{m_0} \cdot r_0^n \times g^{m_1} \cdot r_1^n) \\ &:= (g^{m_0+m_1} \cdot (r_0 \cdot r_1)^n \ mod\ n^2) \\ &:= Enc_{pk}(m_0 + m_1)\,. \end{aligned}$$

A.3.2. SOMEWHAT HOMOMORPHIC ENCRYPTION SCHEME:

The BGV [21] scheme is a leveled HE scheme, and has its security reduced to the Learning with Errors (LWE) problem or its Ring variant (RLWE). Unlike the Paillier scheme, the BGV scheme offers both additive and multiplicative homomorphism albeit under certain conditions. Reference should be made to [21] for further details.

A.3.3. CRYPTOGRAPHIC PROTOCOL

Relevant cryptographic protocols which are utilised to construct our protocol are discussed below.

SECURE MULTIPLICATION PROTOCOL (SMP):

Paillier homomorphic cryptosystem [14] only offers additive homomorphism and no multiplicative homomorphism. We therefore initiate a secure two-party protocol in order to obtain the multiplication of two encrypted values [26]. $Alice$ holds two ciphertexts $Enc_{pk}(m_1)$ and $Enc_{pk}(m_2)$ and requires to compute the product $Enc_{pk}(m_1 \cdot m_2)$.

Bob holds the secret key sk, $Alice$ selects two random numbers r_1, r_2, and computes $Enc_{pk}(m_1 + r_1)$ and $Enc_{pk}(m_2 + r_2)$. Alice sends encryptions of the masked values to Bob. Bob decrypts the ciphertexts and multiplies the results, then encrypts $Enc_{pk}(m_1 \cdot m_2 + m_1 \cdot r_2 + m_2 \cdot r_1 + r_1 \cdot r_2)$ and sends to $Alice$. Finally, $Alice$ can unmask the value to obtain
$Enc_{pk}(m_1 \cdot m_2)$, without having to learn the values of m_1 and m_2 .

SECURE COMPARISON:

As a high level overview, a secure and privacy-preserving comparison protocol takes as inputs a pair of encrypted integer values $Enc_{pk}(a)$, $Enc_{pk}(b)$ from $Alice$. Then she establishes a two-party protocol with Bob who has the decryption key. At the end of the protocol, Bob returns an encryption of a bit $Enc_{pk}(\lambda')$ to $Alice$. $Alice$ further computes on $Enc_{pk}(\lambda')$ to obtain $Enc_{pk}(\lambda)$, where $\lambda, \lambda' \in \{0,1\}$ and the value of b is determined as follows:

$$\lambda := \begin{cases} 0 & \text{if} \quad a < b \\ 1 & \text{if} \quad a \geq b \end{cases} \tag{A.4}$$

In our adopted comparison protocol, $Alice$ does not directly send $Enc_{pk}(a), Enc_{pk}(b)$ to Bob. She however sends $Enc_{pk}(d)$ to Bob, given that $d = 2^{\ell} + r + (2 * a) - (2 * b + 1)$, where r is defined as a random variable in the size of the security paramter, and ℓ is defined as the input bit size. Bob can then decrypt $Enc_{pk}(d)$ and proceed with the two-party computation with the help of $Alice$.

Readers can refer to the paper [20] for more details about the secure comparison protocol.

A.3.4. SECURITY ASSUMPTIONS:

Our secure division protocol is constructed in the semi-honest security model. The channel of communication is assumed to be protected by other network security and cryptographic techniques, thereby considered secure, and there is a non-collusion assumption between $Alice$ and Bob. Our assumption is realistic for a practical setting. For instance when an individual utilises the cloud to compute an operation, the cloud infrastructure might not want to collude with the individual for the sake of their reputation. If the individual were computing on his personal data (encrypted under a third-party key) it then becomes easy to argue that the individual has no incentive for colluding with the cloud services to threaten his privacy. We demonstrate that a user can easily outsource encrypted values to a processing entity such as the cloud to compute division securely, and will receive an encryption of the resulting quotient. Most importantly, our protocol guarantees that the computation was done correctly and the curious processing entity could not infer anymore information from the computation than was expected.

A.3.5. NOTATIONS

Basic notations and their corresponding descriptions as used in the rest of our work are provided in Table A.1.

A

Table A.1: Notations

Notation	Description
$(pk,\ sk)$	Public key, private key pair
$Enc_{pk}(\cdot)$	Encryption algorithm
$Dec_{sk}(\cdot)$	Decryption algorithm
$[\![\cdot]\!]$	Ciphertext
a, b	The operands, a is numerator and b is divisor
ℓ	Upper bound of operands bit size
ℓ'	Input bit size for comparison protocol
q_i	Digit of the resulting quotient at index i
c	Index, ranging from 0 to 10
$\mathcal{K}, \mathcal{K}', \mathcal{K}''$	Vector of bit encryptions
$\mathcal{K}_c$	Ciphertext at index c of vector $\mathcal{K}$
ρ	Digit precision of quotient
n	Plaintext size of encryption scheme
m	Ciphertext bit size
κ	Security parameter, with default value of 80
$\oplus$	Exclusive or
λ'	Bit sent from Bob to $Alice$ after comparison
λ	Bit 1, if $a \geq b$, and bit 0, otherwise
Υ	Secure permutation function
Υ'	The inverse of Υ such that, $A = \Upsilon'(\Upsilon(A))$
$\mathscr{B}$	Number base of computation, default is 10
$EPPCP$	Efficient comparison protocol used
d	The variable $Alice$ sends to Bob to initiate comparison
$\perp$	Empty string
$\overset{c}{\equiv}$	Computationally indistinguishable

A.4. SECURE DIVISION PROTOCOL

The protocol description for our work is presented in this section. We demonstrate how two mistrusting parties $Alice$ and Bob can run a secure two-party protocol with encrypted inputs, to compute the encrypted quotient of the encrypted values, while preserving the privacy of the input variables. Our proposal leverages on a secure and privacy-preserving comparison protocol, which can efficiently compare two encrypted inputs.

A.4.1. PROTOCOL SETTINGS

In our construction, $Alice$ holds two encrypted inputs $Enc_{pk}(a)$, and $Enc_{pk}(b)$, of size ℓ -bits each, while Bob holds the decryption key sk . At the end of the protocol, $Alice$ obtains an encryption of the quotient in fixed-point: $Enc_{pk}(\frac{a}{b})$ while Bob does not learn the value of the quotient, assuming that the comparison protocol is secure. The resulting encrypted fixed-point value will be bounded by a public pre-specified precision value ρ .

The quotient will be a decimal number with ρ digits, $q_0, q_1, \ldots, q_{\rho-1}$. The following conditions are equally necessary for a correct computation of the quotient:

- All operands must be positive integers.
- The relation $a \leq 10 \cdot b$ must hold between the numerator and denominator in order to preserve correctness of the computation. The number 10 represents the number of digits available in the number base in which the processing is performed. For cases where the mentioned relation does not hold, our construction can be adjusted to accommodate such cases.
- As indicated in Subsection A.3.3, the input values to the comparison protocol gets multiplied by 2, followed by the multiplication of the denominator by 10. These operations constitute bit expansion of the original input values. Consequently, for every operand of ℓ bit size, there is an expansion by at most 5 bits. We account for such expansions by defining $\ell' := \ell + 5$, and the comparison protocol can then handle the data expansion to guarantee correctness.

We define $\ell' := \ell + 5$ to clearly distinguishes the input bit size of the comparison protocol ℓ', from the operands bit size ℓ.

Furthermore, the Efficient Privacy-Preserving Comparison Protocol (EPPCP) [20], is described using a hybrid of two additive homomorphic scheme. However, we adopt a slightly modified version of their description, and utilise only a single encryption scheme for clarity and simplicity. This means that we do away with the zero check technique afforded by the DGK [27] scheme, and then replace it with a decryption and a zero check steps respectively. Our approach allows for robustness, because it allows for the bit size of the input to be increased and not limited by the DGK [27] cryptographic scheme.

A.4.2. PROTOCOL DESCRIPTION

Our secure division protocol is described in Protocol 3, using the specification listed in Table A.2. Our construction observes the conditions mentioned in Subsection A.4.1, and then it computes the decimal quotient by iteratively computing the digits of the quotient, until a pre-specified precision ρ. When the operands do not conform to the conditions, they could be homomorphically processed to preserve the required relation. Every round of the division protocol produces an encrypted digit of the operand, by using the EPPCP to compare $Enc_{pk}(b \cdot c) > Enc_{pk}(a)$, $\forall\ c:\ c \in \{1, \ldots, 10\}$. The encrypted result of the EPPCP, which is a binary value is then stored in the corresponding index of the vector $\mathcal{K}$. After the vector $\mathcal{K}$ is constructed, $Alice$ can then use $\mathcal{K}$ to privately compute the desired digit. For any pair of operands $(Enc_{pk}(a), Enc_{pk}(b))$, the first digit of the quotient $Enc_{pk}(q_0)$ obtained from the first round of the division protocol represents the integer part of the quotient. While the subsequent rounds generate the digits of the decimal parts from $Enc_{pk}(q_1), \cdots, Enc_{pk}(q_{\rho-1})$.

We provide further details for the various steps of the protocol.

Step 1: $Alice$ holds two encrypted values $(\llbracket a \rrbracket, \llbracket b \rrbracket)$, which should conform to the conditions in Subsection A.4.1, she sets the value for ρ being the digit precision of the quotient, and sets her counter to $i = 0$. She also initialises the ciphertext variables $\llbracket num \rrbracket$ and $\llbracket den \rrbracket$, and sets them to the numerator and denominator respectively.

A

Table A.2: Protocol Specification

Player	$Alice\,(pk)$	$Bob\,(pk, sk)$
Input	$Enc_{pk}(a)\,, Enc_{pk}(b)$	$\perp$
Output	$Enc_{pk}(q_i)\,; i \in \{0,\ldots,\rho-1\}$	$\perp$
Constraint	$1 \le c \le \mathscr{B};\, a, b \le 2^{\ell};\, a < 10 \cdot b$	

Protocol 3 Division Protocol

1: $Alice$ sets $\rho, i := 0, [\![num]\!] := [\![a]\!]\,, [\![den]\!] := [\![b]\!]$
2: for $c \in \{1\,,\ldots\,,\mathscr{B}\}$
3: compute $[\![den_c]\!] := [\![den \cdot c]\!]$
4: $r_c \leftarrow \{1^{\ell'+\kappa+1}\}$
5: $[\![d]\!]_c := [\![2 \cdot den_c - (2 \cdot num + 1) + r_c + 2^{\ell'}]\!]$
6: $Alice$ applies permutation function $\Upsilon(\cdot)$ to change order of $[\![d]\!]$
7: $Alice$ runs EPPCP with Bob for all values of $[\![d]\!]$
8: Bob returns $[\![\lambda'_c]\!]$ for every run of EPPCP
9: $Alice$ inverts permutation function and obtains: $\mathscr{K} := \Upsilon'(\cdot)$
10: $\mathscr{K} := \{[\![0]\!][\![\lambda_1]\!], [\![\lambda_2]\!], \ldots, [\![\lambda_{10}]\!]\}$
11: $Alice$ construct another vector $\mathscr{K}'$
12: computes values for index as $\mathscr{K}'_c := [\![\lambda_c \oplus \lambda_{c+1}]\!]$
13: $Alice$ computes digit as $[\![q_i]\!] := [\![\sum_{c=1}^{9} (\mathscr{K}'_c \cdot c)]\!]$
14: if $i < (\rho - 1)$
15: set $[\![num']\!] := [\![(num - q_i \cdot b)]\!]$
16: $[\![num]\!] := [\![num' \cdot \mathscr{B}]\!]$
17: $i := i + 1$
18: return to Step 2
19: else END.

Steps 2 - 5: $Alice$ computes 10 values of $[\![d]\!]$ for running the comparison protocol with Bob, where the number 10 represents all decimal digits. $[\![d]\!]_c$ is computed as a comparison of the relation $[\![den \cdot c]\!] > [\![num]\!]$, for value of c ranging from $\{1\,,\ldots\,,10\}$. r_c is a fresh random number for index c .

Step 6: After computing the values of $[\![d]\!]$, $Alice$ permutes the ordering of the ciphertext, in order to obfuscate the ordering when presented to Bob.

Step 7 - 8: $Alice$ initiates a run of the comparison protocol with Bob, and for each $[\![d]\!]_c$ sent to Bob, $Alice$ receives $[\![\lambda'_c]\!]$.

Step 9 - 10: $Alice$ reverses the permutation applied in **Step 6**, in order to re-order the ciphertexts obtained from Bob. This results to the vector of ciphertexts $\mathscr{K}$. Observe that $Alice$ inserts an encryption of zero $[\![0]\!]$, into the first index of the vector $\mathscr{K}$, this is done to extend the size of the vector $\mathscr{K}$ to 1. As a result, all consecutive pair entries of the vector $\mathscr{K}$, can generate 10 entries of the required vector $\mathscr{K}'$.

Step 11 - 12: Then $Alice$ constructs a different vector $\mathscr{K}'$, by homomorphically computing the x-or of every consecutive pair of $[\![\lambda_c]\!]$.

Step 13: $Alice$ can then compute the digit $[\![q_i]\!]$ at position i of the quotient, by first

multiplying every ciphertext at $\mathscr{K}'_c$ by the corresponding scalar c. Then, compute the summation of the results.

Step 14 - 19: If the required precision ρ has been achieved, the algorithm ends, else, the inputs are reset for computation of the next digit, and control returned to **Step 2**.

TOY EXAMPLE:

We illustrate the intuition behind the construction of our protocol with the help of a toy example in Figure A.1. It shows the different steps for computing the division of two encrypted inputs $[\![35]\!]$, $[\![8]\!]$ representing the numerator and denominator respectively, and the quotient presented as a 5 digit precision decimal. In our demonstration, we choose to disregard the natural meanings of the operations (+, *, - >) used, but assume that these operations are implied to work on the encrypted data for simplicity, hence independent of cryptographic scheme. As can be observed, at the end of the protocol, $Alice$ obtains the ciphertexts of the digits of the division, which are $[\![4]\!]$, $[\![3]\!]$, $[\![7]\!]$, $[\![5]\!]$, $[\![0]\!]$, while Bob does not learn any of the digits.

A.4.3. CORRECTNESS

We consider two encrypted positive integers $[\![a]\!]$ and $[\![b]\!]$, where the values $a, b < 2^\ell$ and the quotient $\frac{a}{b} < \mathscr{B}$. Then, rely on the assumption that EPPCP always returns a correct answer for every comparison ($b < a$). To retrieve any digit component of the quotient q_i, we first generate the product of the divisor $[\![b]\!]$ with all possible values of c, which is a trivial case of exhaustive search of all non negative integers smaller than $\mathscr{B}$. If the assumption about the correctness of EPPCP holds, then it must be the case that when the condition $b \cdot c > a$? is True, then:

$$\exists\, c \in [0, ..., 10)\ s.t.\ a \leq b * c\,,\ \text{ and } b * (c-1) < a \tag{A.5}$$

If the secure comparison protocol returns $\lambda_c = 1$ when the comparison is True and $\lambda_c = 0$ otherwise. A vector of bit encryptions $\mathscr{K}$ is generated with the first entry set to an encryption of zero, then we can compute a second vector $\mathscr{K}'$ by xoring every two consecutive entries of the vector $\mathscr{K}$. For cases where the resulting digit of the quotient is a non-zero integer, it can be observed that performing the xor operation will generate ciphertexts of zero-filled vector, with only one position having value $[\![1]\!]$. Multiply each bit encryption with the corresponding index value, and sum the results to extract the index value as the quotient.

When computing the xor operation, we generate a vector of size 10, however, when the digit extraction phase $[\![q_i]\!] = \sum_{c=1}^{9} (\mathscr{K}'_c * c)$ is computed, we only cover for 9 entries of the $\mathscr{K}'$. This is because the operation is a summation, and the value of zero is inconsequential to the final result. All arithmetic operations required within our protocol can be harnessed from the homomorphism properties of the cryptosystem in use. Every run of our protocol produces a digit of the resulting quotient, therefore, after an encrypted digit is obtained, we need to refresh the inputs for the next run. We simple compute $[\![num]\!] := ([\![a]\!] - [\![q_i]\!] * [\![b]\!]) * 10$.

A

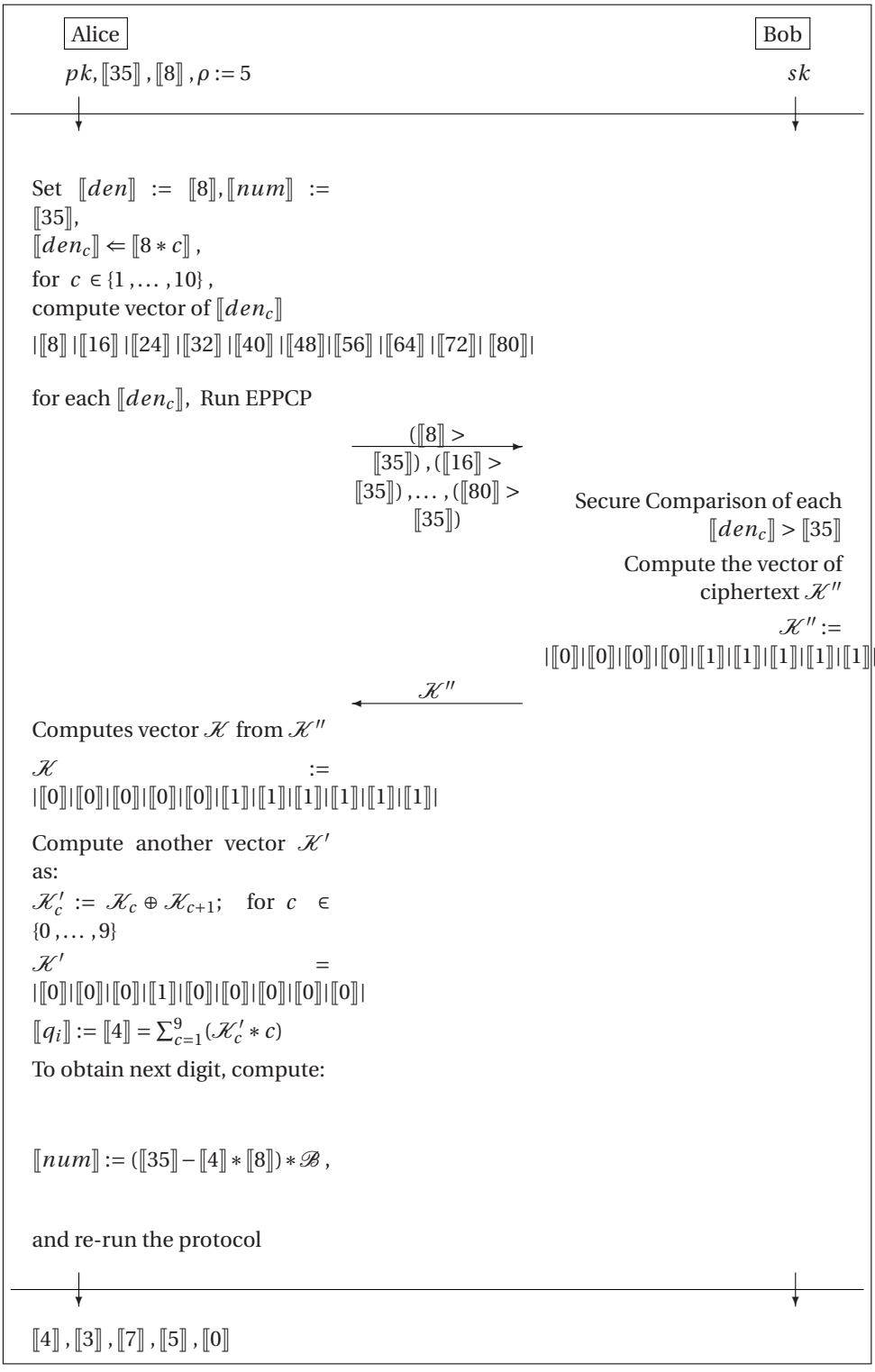

Figure A.1: Division protocol: Toy Example

A.4.4. OPTIMISATION

In a view to reduce the number of rounds and overall complexity of implementation, we can deploy data packing technique. Since we have cryptosystems where bit size of plaintext spaces are often larger than ℓ', we can pack many values of $[\![d_c]\!]$ into a ciphertext and then transmit same to Bob for comparison. Deploying packing will reduce the number of rounds and reduce communication cost, while contributing a negligible computational overhead. Having mentioned that the bit size for the operand values is ℓ', we can proceed to pack w numbers of $[\![d_c]\!]$ in a single ciphertext. To compute w, we have that $w = \lfloor \frac{n}{\kappa+\ell'+1} \rfloor$ where κ is the security parameter. Therefore, for the default implementation of EPPCP where Paillier cryptosystem is used for the packing, we have $\kappa = 80, \ell' = 30$, and $n = 2048$ being the plaintext size of Pallier. Just as expected, the amount of data to be packed decreases proportionally as input size ℓ increases. Deploying packing reduces the number of decryption required. It should be mentioned that decryption costs are computationally expensive and often outweighs the computational cost of other homomorphic operations.

Additionally, when $Alice$ is processing the value $[\![d_c]\!]$ for the comparison protocol, she computes the encryption of $[\![b * c]\!]\ \forall\ c\ \in \{1, 2, \dots, 10\}$. The encrypted values can be reused over the course of computing the encrypted digits $[\![q_i]\!]$, and we can avoid the extra homomorphic additions and scalar multiplications associated with that computation.

We can reduce the cost needed for computation of $[\![\sum_{c=1}^{9}(\mathcal{K}'_c \oplus \mathcal{K}'_{c+1}) * c]\!]$. Rather than computing the xor as $[\![\sum_{c=1}^{9}(\mathcal{K}'_c + \mathcal{K}'_{c+1} - 2 * \mathcal{K}'_c * \mathcal{K}'_{c+1}) * c]\!]$, we process the encrypted values as $[\![\sum_{c=1}^{9}(\mathcal{K}'_{c+1} - \mathcal{K}'_c) * c]\!]$. It follows from the fact that due to the special case of the vector $\mathcal{K}'$, one can compute the expression $\mathcal{K}'_c \oplus \mathcal{K}'_{c+1}$ as either $\mathcal{K}'_c + \mathcal{K}'_{c+1} - 2 * \mathcal{K}'_c * \mathcal{K}'_{c+1}$ or computed as $\mathcal{K}'_{c+1} - \mathcal{K}'_c$. On a close observation of both methods for computing $\mathcal{K}'_c \oplus \mathcal{K}'_{c+1}$, the former requires at least 2-additions, 1-multiplication and 1-subtraction, while the latter requires a single subtraction. We refer readers to the original paper [20] for further details.

A.5. COMPLEXITY ANALYSES

Here, we show the complexity for various components that contribute to the division protocol. We present a tabular representation for the costs contributed by each operation during the run of our protocol in different modes. In Table A.3, we present the

Table A.3: Computation cost for a single EPPCP

	Alice	**Bob**	**Total**
Encryption	$\ell' + 3$	$\ell' + 2$	$2\ell' + 5$
Decryption	0	$\ell' + 1$	$\ell' + 1$
Scalar Mult.	$\ell' + 2$	0	$\ell' + 2$
Multiplication	0	0	0
Addition	$\ell' + 1$	0	$\ell' + 1$
Subtraction	3	0	3

computation cost incurred for every completed comparison of two integers. The table

A

contains only the homomorphic operations because other non-homomorphic operations contribute negligible cost to the protocol. The decryption operation is the most expensive of all the listed operations. It can be seen that the cost of all operations grow linearly in the bit size of the input variables. In Table A.4, we show the communication cost of executing a single round of the comparison protocol.

Table A.4: Communication cost for a single EPPCP

	Alice	**Bob**	**Total**
Sent (bits)	$m(\ell'+1)$	$m(\ell'+2)$	$m(2\ell'+3)$
Received (bits)	$m(\ell'+2)$	$m(\ell'+1)$	$m(2\ell'+3)$

The SMP requires 4, 2, 5 and 2 units of encryption, decryption, addition and scalar multiplication respectively.

To compute a single digit of the quotient, the division protocol requires 8ρ, $10\rho-1$, $10\rho-1$, $\rho-1$ and 10ρ units of addition, subtraction, scalar multiplication, secure multiplication and EPPCP respectively. This shows that the complexity of the division protocol is mostly dependent on the comparison protocol. Also, as can be seen in Table A.5, the computation cost scales linearly with both the bit length of the operands and precision of the quotient. The complexity of computing a quotient is equally dependent on the precision of the quotient, represented as ρ. Furthermore, there is no extra communication cost associated with the division protocol excluding that contributed by the comparison protocol. This is the case because there is no interaction between $Alice$ and Bob for computing the digits, except for that incurred during secure multiplication.

Table A.5: Computation cost for a quotient

Enc.	**Dec.**	**Add.**	**Sub.**	**Mult.**	**Sc. Mult.**
$20\ell'\rho+50\rho$	$10\ell'\rho+10\rho$	$10\ell'\rho+18\rho$	$10\rho+2$	$\rho-1$	$10\ell'\rho+30\rho+1$

A.6. SECURITY AND PRIVACY ANALYSES

In this section we present the security and privacy arguments for which our construction is secure and privacy-preserving under the honest-but-curious security model. We adopt the universal composability (UC) security framework [22, 28–31], due to it being more expressive and suitable for a two-party computation protocols. The idea would be to present a comparison between a real-world construction of our protocol against an ideal-world version of the same protocol. We conclude that a real-world protocol Π, realizes an ideal-world protocol $\mathscr{F}$, if an adversary $\mathscr{A}$ in the real-world protocol does not gain any more advantage than an adversary $\mathscr{S}$ in the ideal-world protocol. Note that in the ideal setting of the protocol ($\mathscr{F}$) , security and privacy are guaranteed and no adversary should be able to compromise the protocol. There exists an environment $\mathscr{E}$, who provides inputs to parties and is able to observe outputs from parties as well as know corrupted parties. We say that Π securely realizes $\mathscr{F}$, if $\mathscr{E}$ cannot distinguish a real-world protocol running with adversary $\mathscr{A}$, from an ideal-world protocol running with simulator $\mathscr{S}$.

Our proof is presented in two parts. This is important because we utilise two subprotocols (secure comparison, secure multiplication[1]) in constructing our secure division protocol. We provide security proof for one of the sub-protocols using simulation based technique, and finally compose the security using UC framework. **Negligible function**[31]: A function $\mu(\cdot)$ is negligible if for every positive polynomial $p(\cdot)$ and all sufficiently large $\kappa \in \mathbb{N}$, it holds that $\mu(\kappa) < 1/p(\kappa)$.

Computational Indistinguishability[31]: Given that $a \in \{0,1\}^*$ and κ is security parameter, let $X = X(a,\kappa)$ and $Y = Y(a,\kappa)$ be two probability ensembles. X and Y are said to be computationally indistinguishable, denoted by $X \overset{c}{\equiv} Y$, if for every non-uniform polynomial-time algorithm D there exists a negligible function $\mu(\cdot)$ such that,

$$|\Pr[D(X(a,\kappa)) = 1] - \Pr[D(Y(a,\kappa)) = 1]| \leq \mu(\kappa)\,. \tag{A.6}$$

Definition of Security[31]: Let $f = (f_A, f_B)$ be a PPT functionality and let π be a two-party (by A and B) protocol for computing f. f_A, f_B denote the results corresponding to parties A and B respectively on running f. The view of the party $i \in \{A,B\}$ during the execution of π on input (a,b) and security parameter κ is denoted by , $\mathbf{view}_i^{\pi}(a,b,\kappa) := (w, r^i; m_1^i, \ldots, m_t^i)$, where $w \in (a,b), r^i$ equals the content of party i's internal random tape, and m_j^i represents the jth message received.

The output of party i during the execution of π on the inputs (a,b) with security parameter κ is denoted by , $\mathbf{output}_i^{\pi}(a,b,\kappa)$ and can be computed from its own view of the execution. The join output of both parties is denoted by , $\mathbf{output}^{\pi}(a,b,\kappa) = (\mathbf{output}_A^{\pi}(a,b,\kappa), \mathbf{output}_B^{\pi}(a,b,\kappa))$.

We say that π securely computes f in the presence of semi-honest adversaries if there exists PPT algorithms $\mathscr{S}_A$ and $\mathscr{S}_B$ such that:

$$\{\mathscr{S}_A(1^{\kappa}, a, f_A(a,b)), f(a,b)\} \overset{c}{\equiv} \{(\mathbf{view}_A^{\pi}(a,b,\kappa), \mathbf{output}^{\pi}(a,b,\kappa))\}\,. \tag{A.7}$$

$$\{\mathscr{S}_B(1^{\kappa}, b, f_B(a,b)), f(a,b)\} \overset{c}{\equiv} \{(\mathbf{view}_B^{\pi}(a,b,\kappa), \mathbf{output}^{\pi}(a,b,\kappa))\}\,. \tag{A.8}$$

In order to provide a clear proof of the comparison protocol as adopted in this work, we first present the internal working of the protocol. This will help the reader to understand the proof presented below. The two parties are A and B.

1. A and B are given security parameter κ, pk and input bit length ℓ . B is given sk .
2. A is given $[\![a]\!]$, $[\![b]\!]$ each of size ℓ bits.
3. A generates $r_A \leftarrow \{0,1\}^{\ell+\kappa}$ and computes $[\![d]\!] := [\![2^{\ell} + r_A + 2a - 2b - 1]\!]$ and sends $[\![d]\!]$ to B.
4. B decrypts $[\![d]\!]$ and computes $d' := d \bmod 2^{\ell}$, and $d'' := \lfloor \frac{d}{2^{\ell}} \rfloor$.
5. For $i \in \{0,\ldots,\ell-1\}$, B computes $t_i = d'_i + \sum_{j=i+1}^{i-1} 2^j * d'_j$. and sends $[\![d'']\!], [\![t_0]\!], \ldots, [\![t_{\ell-1}]\!]$ to A .
6. A computes $\hat{r}_A := r_A \bmod 2^{\ell}, s \leftarrow \{-1,1\}$; For $i \in \{0,\ldots,\ell-1\}$,
   $h_i \leftarrow \mathbb{Z}_{\kappa}^*; v_i := s - \hat{r}_{Ai} - \sum_{j=i+1}^{\ell-1} 2^j * \hat{r}_{Aj}; [\![c_i]\!] := [\![v_i + t_i]\!]; [\![e_i]\!] := [\![c_i * h_i]\!]$.
   And sends all encrypted values of $[\![e_i]\!]$ to B.
7. B checks if any $[\![e_i]\!]$ decrypts to zero, and sets $\lambda' := 1$, and if no zero value is found $\lambda' := 0$. B returns $[\![\lambda']\!]$ to A.

[1] The secure multiplication protocol is widely used in literature and has been proven secure, so we will not be providing that proof in this work.

A

8. On receiving $[\![\lambda']\!]$, A sets the variable $[\![x]\!] := [\![\lambda']\!]$ if $s = 1$, else set $x := [\![1-\lambda']\!]$ if $s = -1$.
9. Finally, A computes $[\![\lambda]\!] := [\![d'' - \lfloor \frac{r_A}{2^\ell} \rfloor - x]\!]$.

The comparison protocol π securely and privately computes the comparison functionality $f(([\![a]\!], [\![b]\!]), \perp) = ([\![\lambda]\!], \perp)$ in the presence of any honest-but-curious PPT adversary.

The aim here is to show that the view of a PPT adversary $\mathscr{A}$ in the real-world execution of the protocol π, is computationally indistinguishable from the view of a simulator $\mathscr{S}_i$ for $i \in \{A, B\}$ in the ideal world execution of the protocol f. Let us assume that party A has been compromised by an adversary $\mathscr{A}$. $\mathscr{S}_A$ is provided with the inputs and outputs of party A, and is required to simulate the view:

1. $\mathscr{S}_A$ is provided with $[\![a]\!]$ and $[\![b]\!]$, and he generate $r_{SA} \leftarrow \{0,1\}^{\ell+\kappa}$ and computes $[\![d_{SA}]\!] := [\![2^\ell + r_{SA} + 2a - 2b - 1]\!]$ and sends $[\![d_{SA}]\!]$ to B.
2. B follows the protocol and sends $[\![d''_{SA}]\!], [\![t_{SA0}]\!], \ldots, [\![t_{SA\ell-1}]\!]$ to A.
3. $\mathscr{S}_A$ continues with the protocol, then generates and sends all encrypted values of $[\![e_{SAi}]\!]$ to B.
4. B responds with $[\![\lambda'_{SA}]\!]$ to A.
5. $\mathscr{S}_A$ completes the protocol and generates $[\![\lambda_{SA}]\!]$

From above, the simulated view of $\mathscr{S}_A$ can be expressed as:

$$\mathscr{S}_A(([\![a]\!], [\![b]\!]), \kappa) := (1^\kappa, r_{SA}, ([\![a]\!], [\![b]\!]), [\![d''_{SA}]\!], [\![t''_{SA0}]\!], \ldots, [\![t''_{SA\ell-1}]\!], [\![\lambda'_{SA}]\!]) \tag{A.9}$$

In contrast, the view of party A is presented as:

$$\mathbf{view}^\pi_A(([\![a]\!], [\![b]\!]), \kappa) := (1^\kappa, r_A, ([\![a]\!], [\![b]\!]), [\![d'']\!], [\![t''_0]\!], \ldots, [\![t''_{\ell-1}]\!], [\![\lambda']\!]) \tag{A.10}$$

We can conclude from Equation (A.10, A.9) that

$$\mathscr{S}_A(([\![a]\!], [\![b]\!]), \kappa) \overset{c}{\equiv} \mathbf{view}^\pi_A(([\![a]\!], [\![b]\!]), \kappa). \tag{A.11}$$

For any PPT distinguisher D,

$$\begin{aligned} &Pr[(1^\kappa, r_{SA}, ([\![a]\!], [\![b]\!]), [\![d''_{SA}]\!], [\![t''_{SA0}]\!], \ldots, [\![t''_{SA\ell-1}]\!], [\![\lambda'_{SA}]\!]), ([\![\lambda_{SA}]\!]) = 1] \\ -&Pr[(1^\kappa, r_A, ([\![a]\!], [\![b]\!]), [\![d'']\!], [\![t''_0]\!], \ldots, [\![t''_{\ell-1}]\!], [\![\lambda']\!]), ([\![\lambda]\!]) = 1] \leq \frac{1}{\mu(\kappa)} \end{aligned} \tag{A.12}$$

Finally, for any semantically secure encryption scheme, Eq. A.12 will hold and our protocol π remains secure. And that completes the first part of the proof.

If party B is compromised, we construct a similar proof, with $\mathscr{S}_B$.

$$\mathscr{S}_B((\perp), \kappa) := (1^\kappa, r_{SB}, [\![d]\!], d, e_{SB0}, \ldots, e_{SB\ell-1}, [\![e_{SB0}]\!], \ldots, [\![e_{SB\ell-1}]\!]). \tag{A.13}$$

Again in contrast, the view for party B is presented as:

$$\mathbf{view}^\pi_B((\perp), \kappa) := (1^\kappa, r_B, [\![d]\!], d, e_0, \ldots, e_{\ell-1}, [\![e_0]\!], \ldots, [\![e_{\ell-1}]\!]). \tag{A.14}$$

For any PPT distinguisher D,

$$Pr[(1^\kappa, r_{SB}, [\![d]\!], d, e_{SB0}, \ldots, e_{SB\ell-1}, [\![e_{SB0}]\!], \ldots, [\![e_{SB\ell-1}]\!]), (\perp) = 1]$$
$$-Pr[(1^\kappa, r_B, [\![d]\!], d, e_0, \ldots, e_{\ell-1}, [\![e_0]\!], \ldots, [\![e_{\ell-1}]\!]), (\perp) = 1] \leq \frac{1}{\mu(\kappa)} \quad \text{(A.15)}$$

In order to extend the proof to our division protocol, we denote the division protocol as Π. We say that Π securely realizes the ideal functionality $\mathcal{F}$, and that Π calls the subprotocol (comparison protocol) f. We represent the call of Π to f as the hybrid Π^f, and claim that this must be a secure construction being that f is an ideal protocol. If π securely realizes f, then, without loss of generality, we can conveniently replace the call to f, with the call to π. We then conclude that Π^π securely realizes Π^f in the presence of any PPT distinguisher $\mathcal{E}$. The protocol Π is a secure construction[2].

If $Alice$ is expected to learn the decrypted values of the computation, then she can run a secure decryption protocol with Bob, and thereafter $Alice$ can learn as much information as she would have learned if this protocol were to be replaced with a trusted and unbiased third-party. This means that no information learned by $Alice$ is a consequence of deploying the secure division protocol, but as an natural information leak from the computation. For instance, if the value of the quotient is $[\![1]\!]$, then it must be the case that both $[\![a]\!]$ and $[\![b]\!]$ encrypt the same value. If the quotient is $[\![0]\!]$, then $[\![a]\!] = [\![0]\!]$ and this can only happen with probability $2^{-\ell}$, assuming uniform selection of operands. The same follows for the case where the quotient only has an integer part with the decimal component all values of zero; then $Alice$ learns that the numerator is divisible by the denominator. And finally, if the quotient is an even integer, $Alice$ can conclude for certainty that the value of $[\![a]\!]$ was an even integer.

A.7. IMPLEMENTATION RESULTS

We demonstrate that our protocol is cryptosystem independent, and show performance of our construction when implemented with different schemes. Therefore, we provide implementation for two scheme, first with SeComLib [32] which implements the Paillier cryptosystem and secondly with HeLib [33], which implements the BGV cryptosystem. All our implementations were carried out on an Intel core 2 Quad @ 2.66GHZ machine, running Ubuntu 14.04 LTS. The default security parameter $\kappa = 80$, while the value for ℓ varies with the cryptosystem to allow for robustness. Figures A.2. and A.3. provide a visual that explains how each homomorphic operation contributes to the total runtime of the protocol, depending on the adopted cryptosystem. Although both cryptosystems can be optimised the achieve better performance, the Figures show that while to compute a single digit of the quotient, Paillier cryptosystem requires about 2 $seconds$. While the BGV scheme requires about 20 $seconds$. Figure A.2. equally show that in the Paillier scheme, decryption cost overwhelms the total cost of executing the protocol, while Figure A.3. shows that encryption cost constitutes the most significant part of the total execution time. We can conclude that in the Paillier scheme implementation, the decryption cost is associated with the zero-check step within the comparison protocol, and if this step could be replaced by a much efficient techniqe, then the total execution

[2] Recall that other phases of the division protocol are non-interactive and are computed homomorphically

A

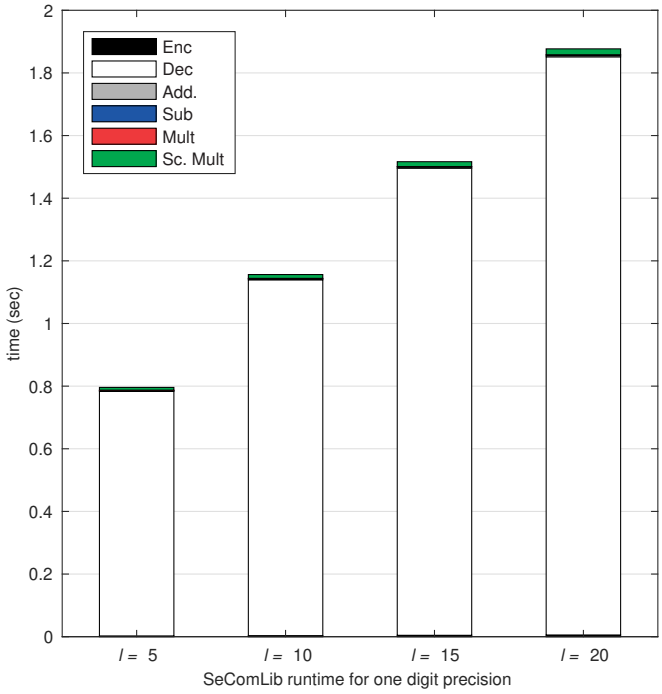

Figure A.2: Runtime of the protocol with seComLib.

time can be further optimised. There is evidently more incentive to adopt the Paillier implementation, as this provides for a better runtime relative to that of the BGV scheme. Secondly, since the ciphertext size of of the Paillier scheme is 4096 bits, as against the more than 2 Mbits required for one ciphertext of the BGV scheme. Although the BGV scheme offers non-interactive homomorphic multiplication, which is not available with the Paillier scheme, the overall advantage of the BGV scheme is still negligible compared to the storage and processing costs associated with implementing the scheme.

Table A.6: Runtime for homomorphic operations in seconds

	Enc.	**Dec.**	**Add.**	**Sub.**	**Mult.**	**Sc. Mult.**
SeComLib	1×10^{-5}	0.0071	1.9×10^{-5}	1.2×10^{-4}	0.01445	6.7×10^{-5}
HELib	0.0381	0.0143	2.4×10^{-4}	6.1^{-4}	0.0683	3×10^{-4}

Table A.6 presents individual operation runtime for the two cryptosystems implementation. The table helps to further appreciate the details in Figure A.2 and Figure A.3

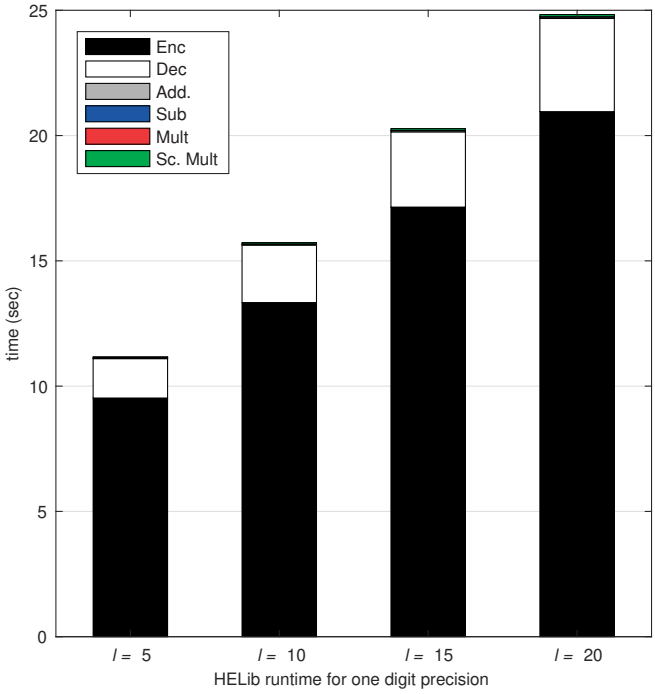

Figure A.3: Runtime of the protocol with HELib.

A.8. CONCLUSION

Secure division of encrypted values is a secure alternative for providing confidentiality and privacy for outsourced data. In this paper, we present an efficient and secure division protocol which takes as input, two homomorphically encrypted operands, and outputs the fixed-point quotient up to a predefined precision. Our construction guarantees the privacy and security of the operands, under a non-collusion assumption. The protocol is simple, robust, and can scale efficiently with the length of operands bit size as well as the digit precision of the quotient. On an average desktop computer, our solution computes the division of two encrypted operands in less than 2 seconds. A secure comparison protocol is leveraged for our proposal, and the secure comparison component can be replaced whenever a more efficient comparison protocol is available. We provide security analyses, complexity analyses and implementation to demonstrate the feasibility and robustness of our proposal. Implementation results are also provided to show performance using two types of homomorphic cryptographic schemes.

A

REFERENCES

[1] M. Belenkiy, M. Chase, C. C. Erway, J. Jannotti, A. Küpçü, and A. Lysyanskaya, *Incentivizing outsourced computation,* in *Proceedings of the 3rd international workshop on Economics of networked systems* (ACM, 2008) pp. 85–90.

[2] J. Loftus and N. P. Smart, *Secure outsourced computation,* in *International Conference on Cryptology in Africa* (Springer, 2011) pp. 1–20.

[3] L. Wei, H. Zhu, Z. Cao, X. Dong, W. Jia, Y. Chen, and A. V. Vasilakos, *Security and privacy for storage and computation in cloud computing,* Information Sciences **258**, 371 (2014).

[4] G. Danezis and E. De Cristofaro, *Fast and private genomic testing for disease susceptibility,* in *Proceedings of the 13th Workshop on Privacy in the Electronic Society* (ACM, 2014) pp. 31–34.

[5] C. Ugwuoke, Z. Erkin, and R. L. Lagendijk, *Privacy-safe linkage analysis with homomorphic encryption,* in *Signal Processing Conference (EUSIPCO), 2017 25th European* (IEEE, 2017) pp. 961–965.

[6] B. Adida, *Helios: Web-based open-audit voting.* in *USENIX security symposium*, Vol. 17 (2008) pp. 335–348.

[7] K. Lauter, A. López-Alt, and M. Naehrig, *Private computation on encrypted genomic data,* in *Progress in Cryptology-LATINCRYPT 2014* (Springer, 2014) pp. 3–27.

[8] W. Lu, Y. Yamada, and J. Sakuma, *Efficient secure outsourcing of genome-wide association studies,* in *Security and Privacy Workshops (SPW), 2015 IEEE* (IEEE, 2015) pp. 3–6.

[9] Z. Erkin, *Private data aggregation with groups for smart grids in a dynamic setting using crt,* in *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on* (IEEE, 2015) pp. 1–6.

[10] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, *Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption,* IEEE transactions on parallel and distributed systems **24**, 131 (2013).

[11] A. Shamir, *How to share a secret,* Communications of the ACM **22**, 612 (1979).

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

[13] R. L. Rivest, L. Adleman, and M. L. Dertouzos, *On data banks and privacy homomorphisms,* Foundations of secure computation **4**, 169 (1978).

[14] P. Paillier, *Public-key cryptosystems based on composite degree residuosity classes,* in *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding* (1999) pp. 223–238.

[15] C. Gentry, *A fully homomorphic encryption scheme*, Ph.D. thesis, Stanford University (2009).

[16] T. Veugen, *Encrypted integer division.* in *WIFS* (2010) pp. 1–6.

[17] M. Dahl, C. Ning, and T. Toft, *On secure two-party integer division,* in *Financial Cryptography and Data Security* (Springer, 2012) pp. 164–178.

[18] M. Franz, B. Deiseroth, K. Hamacher, S. Jha, S. Katzenbeisser, and H. Schröder, *Secure computations on non-integer values,* in *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on* (IEEE, 2010) pp. 1–6.

[19] T. Veugen, *Encrypted integer division and secure comparison,* International Journal of Applied Cryptography **3**, 166 (2014).

[20] M. Nateghizad, Z. Erkin, and R. L. Lagendijk, *An efficient privacy-preserving comparison protocol in smart metering systems,* EURASIP Journal on Information Security **2016**, 1 (2016).

[21] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, *(leveled) fully homomorphic encryption without bootstrapping,* in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (ACM, 2012) pp. 309–325.

[22] R. Canetti, A. Cohen, and Y. Lindell, *A simpler variant of universally composable security for standard multiparty computation,* in *Annual Cryptology Conference* (Springer, 2015) pp. 3–22.

[23] P. Bunn and R. Ostrovsky, *Secure two-party k-means clustering,* in *Proceedings of the 14th ACM conference on Comp. and comm. security* (ACM, 2007) pp. 486–497.

[24] O. Catrina and A. Saxena, *Secure computation with fixed-point numbers,* in *Int. Conf. on Financial. Cryptography and Data Sec.* (Springer, 2010) pp. 35–50.

[25] M. Aliasgari, M. Blanton, Y. Zhang, and A. Steele, *Secure computation on floating point numbers.* in *NDSS* (2013).

[26] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, *Generating private recommendations efficiently using homomorphic encryption and data packing,* Information Forensics and Security, IEEE Transactions on **7**, 1053 (2012).

[27] I. Damgard, M. Geisler, and M. Kroigard, *Homomorphic encryption and secure comparison,* International Journal of Applied Cryptography **1**, 22 (2008).

[28] R. Canetti, *Universally composable security: A new paradigm for cryptographic protocols,* in *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on* (IEEE, 2001) pp. 136–145.

[29] J. Groth, *Evaluating security of voting schemes in the universal composability framework,* in *International Conference on Applied Cryptography and Network Security* (Springer, 2004) pp. 46–60.

[30] R. Küsters and M. Tuengerthal, *The iitm model: a simple and expressive model for universal composability.* IACR Cryptology EPrint Archive **2013**, 25 (2013).

[31] Y. Lindell, *How to simulate it–a tutorial on the simulation proof technique,* in *Tutorials on the Foundations of Cryptography* (Springer, 2017) pp. 277–346.

[32] *SeComLib Secure Computation Library*, Cyber Security Group, TU Delft (2010), `http://cybersecurity.tudelft.nl/content/secomlib`.

[33] S. Halevi and V. Shoup, *Algorithms in helib,* in *International Cryptology Conference* (Springer, 2014) pp. 554–571.

B

APPENDIX ECONOMY: ENSEMBLE COLLABORATIVE LEARNING USING MASKING

In a society where digital data has become ubiquitous and has been projected to continue in this trajectory for the foreseeable future, machine learning has become a dependable tool to aid in analyzing these big dataset. However, where the data or machine learning algorithms are considered to be privacy-sensitive, one is then faced with the challenge of preserving the utility of machine learning in a privacy-preserving setting. In this paper, we focus on a use case where decentralized parties have privately owned machine learning algorithms, and would want to jointly generate a public model while not violating the privacy of their individual models, and data. We present ECONoMy: a privacy-preserving protocol that supports collaborative learning using an ensemble technique. Set in an honest-but-curious security model, ECONoMy is light-weight and provides efficiency and privacy in settings with large participant such as with IoT devices.

Parts of this chapter have been published as:
(1) Van De Kamp, L., Ugwuoke, C., & Erkin, Z. (2019, September). ECONoMy: Ensemble Collaborative Learning Using Masking. *In 2019 IEEE 30th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC Workshops) (pp. 1-6). IEEE.*

B.1. INTRODUCTION

The rapid growth of digital data over the past few decades has presented an interesting and peculiar challenge of big data [1, 2]. Although the most common challenge posed by this event can be said to be that of storage, perhaps, a more critical challenge is the processing of huge data especially when faced with privacy-sensitive data, such as medical data [3]. Huge dataset presents increasing complexity for classical algorithms to navigate through, so, it has become common to adopt machine learning (ML) in order to analyze data which would otherwise be both complex and almost impossible to analyze. More so, the industry-acceptance of machine learning has become rife, due to its ability to extract actionable information from a large dataset [3–7].

Organizations and their customers want to benefit from improved services, offering higher utility and increased productivity. But recent data leaks incite a widespread awareness of privacy, as they highlight the risks accompanying the collection of large amounts of data needed to provide the desired benefits. Making organizations and even users more circumspect in sharing sensitive data with third parties.

Witten and Frank [8] define machine learning as finding and describing structural patterns within data, which offers additional knowledge on the data, and can help make predictions on new samples. The different data points, or samples, used to retrieve such structural insights can either be labeled, which refers to the concept of supervised learning, or unlabeled, also known as unsupervised learning.

Theodoridis and Koutroumbas [9] define supervised learning as, designing a classifier by exploiting a priori known information, namely the labels corresponding to the available training data. Unsupervised learning, on the other hand, can be used without the presence of class labels. A given set of feature vectors attributes extracted from the training data, is used to determine underlying similarities or clusters, thereby grouping 'similar' items together [9]. Semi-supervised learning exists in between these two types, having access to a set of patterns without their corresponding class, and to a subset for which the class is known. The unlabeled data can then be used to obtain additional information about the general structure belonging to the data at hand [9].

Collaborative learning is one approach where two or more entities contribute their individual resources and skill in order to learn. Hence, they entities capitalize on the unit knowledge from participant to achieve a global knowledge. In the context of machine learning, this would mean that multiple data owners cooperate to jointly generate a model. However, in a privacy-sensitive setting, the participants aim to limit the privacy loss of their collected training data to other collaborators.

B.1.1. ADVERSARIAL MACHINE LEARNING

AML is a growing research area that focuses on training and use of machine learning classifiers in adversarial environments [10]. This field assumes that the environment contains a potential adversary who can be defined as one's opponent in a contest, conflict or dispute [11]. Adversaries conduct different types of attacks on the training or inference phases of machine learning to reach a particular objective. Huang et al. [12] have proposed a taxonomy for the research field in which they define these different types of attacks. The taxonomy classifies attacks on three different criteria: 1) the type of influence exerted, 2) the resulting security violation, and 3) the specificity of the at-

tack. The influence that an attack can be either causative or exploratory, where the first can actively adapt the training data whereas the latter attempts to obtain information about the training data or underlying model. The second requirement evaluates the impact a successful attack can have. An attack could lower the integrity of the system by allowing malicious samples to be labeled as benign, degrade the availability by increasing the number of wrong classifications in general, or by extracting privacy-sensitive information belonging to the learner. Finally, an attack can focus on a particular misclassification (or target), or any misclassification (all possible targets) determining the specificity of the aim of the attack. A more generic classification of attacks distinguishes between poisoning and evasion attacks. Poisoning attacks focus on the training phase, whereas evasion attacks focus on the inference phase of machine learning. The work presented in this paper mainly focuses on the loss of privacy of the learners participating in a collaborative learning protocol, exercised to train a globally shared classifier. Using the previously described terminology, our focus can be described as exploratory attacks, extracting privacy-sensitive information focused on either all participants or a specific victim.

B.1.2. CROWDML

CrowdML, as proposed by Hamm et al. [13] follows the semi-supervised approach to machine learning thereby bounding the privacy loss of the final released model. Nevertheless, this approach does trust a central entity in such a way that the participants transfer their models from the participants without taking into account potential privacy loss this can cause. Nevertheless, our ECONoMy approach aims to be used in similar situations where a lot of small devices are used to generate an ensemble of the underlying data.

In this paper, we confine our setting to a scenario where a large number of participants, such as in an IoT setting wish to utilize a decentralized architecture to jointly generate a global model from their individual private models. We focus on semi-supervised learning using ensemble machine learning. The local models are securely generated from private dataset, while the interactive processes are described in the semi-honest security model.

B.2. BACKGROUND

B.2.1. ENSEMBLE LEARNING

Ensemble learning is a form of supervised learning that uses an 'ensemble' of different classifiers rather than a single model. An ensemble is a collection of a (finite) number of predictors that are trained independently for the same task, after which their predictions are combined [14]. Lior Rokach [15] refers to an ensemble as a weighted combination of the individual classifier opinions to obtain a classifier that outperforms every single one.

The employed classification technique of the ensemble is used to determine the class label corresponding to an unlabeled sample. An overview of combination methods for ensemble learning has been discussed by Lior Rokach [15]. Due to space limitation, we only concentrate on one of the techniques known as “majority vote" to perform the label determination based on the input of individual classifiers.

Majority vote technique which uses weighing methods is also known as the plurality vote (PV) or basic ensemble method (BEM) where classification results in the class with the most votes. The formula is expressed as
$class(x) = \underset{c_i \in\, dom(y)}{\arg\max} \left(\sum_k g\left(y_k(x), c_i\right) \right)$, where $y_k(x)$ is the classification of the k'th classifier and $g(y,c) := 1$ if $y = c$ else $g(y,c) := 0$.

B.2.2. COLLABORATIVE LEARNING

The primary focus of collaborative learning is on the cooperation of multiple data sources working together in generating a model. Figure B.1 illustrates the fact that five participants jointly provide inputs to generate a globally known model Ω_G.

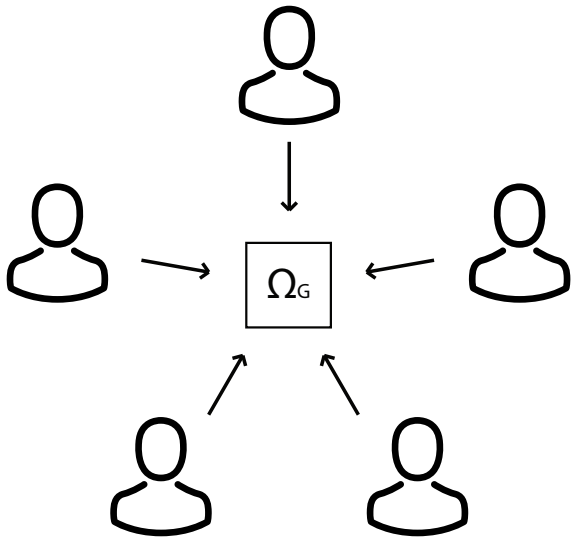

Figure B.1: A visualization of the cooperative nature of collaborative learning

In a collaborative setting, the original training samples remain at its local host, a globally known model is constructed using distributed techniques that aggregate derivatives of the locally acquired knowledge. Such an approach can work in several ways: by either sharing a subset of updated model parameters, as described in the case of a neural network by [16] and [17], by sharing complete models, or by sharing predictions on specific items contained in a public dataset as done in [18].

B.2.3. ADVERSARIAL EXAMPLES

An adversarial example is a modified data sample that is misclassified by a classifier [19], allowing an adversary to trick a model to for example classify malicious traffic as benign. Usually, a perturbation is applied to a valid sample to achieve this and have the classifier decision disagree with what a human would see it as [20]. Such a perturbation can be specifically designed for a single model, a particular target class, or universal

applications. The latter, proposed by Moosavi-Dezfooli et al. [21], computes the perturbation values in such a way that they can convert a diverse range of images into adversarial examples. The creation of an adversarial example can on the other hand already be achieved by the alteration of a single pixel as shown by Su et al. [22]. Kurakin et al. [19] have shown that the concept of adversarial examples also transfers to the physical world by physically printing the generated examples.

Privacy The attacker can compromise privacy by obtaining the underlying model or the information derived from the training data. Thus, attacks and defenses for privacy focus on preventing an adversary from obtaining either. The most prevalent attacks that can harm privacy are: membership inference attacks [23], model inversion attacks [24], model extraction attacks [25], and GAN attacks [17].

Shokri et al. [23] have proposed a membership inference attack that given 1) a data record δ_1 and 2) black-box access to a model m_1, one can determine whether δ_1 was in the training dataset on which model m_1 has been trained. Federikson et al. [24] introduced a model inversion approach abusing confidence levels that accompany label predictions to extract information of the used training data. Tramer et al. [25] propose a method to learn a close approximation of the original model function of a black-box model.

B.2.4. CRYPTOGRAPHIC PRIMITIVES

Additive Secret Sharing We use a variant of the additive secret sharing procedures employed by Kursawe et al. [26], and Gracia et al. [27]. Their approach uses the assignment of leaders and aggregators, unlike ours.

Instead, every individual starts out by generating m random numbers R_{p_i} for $i \in \mathcal{I}$ (defined in Table B.1), once for each item that is to be labeled. These random numbers are then divided into n random shares, one for each of the participants, such that the shares sum up to the original random value. The following example assumes there is only one item, $m = 1$:

$$\text{Alice (1): } R_1 = r_{11} + r_{12} + r_{13} \mod p \tag{B.1}$$

$$\text{Bob (2): } R_2 = r_{21} + r_{22} + r_{23} \mod p \tag{B.2}$$

$$\text{Charlie (3): } R_3 = r_{31} + r_{32} + r_{33} \mod p \tag{B.3}$$

where p is a large prime. Alice then computes her masking value using the following function: $M_1 = \sum_{i=1}^{n} r_{i1} - R_1$

By deducting the original random value, we can be convinced that the final parts that are used in the masking will cancel out to zero and thus leave us with an accurate aggregate of the, to be hidden, messages.

Diffie-Hellman Diffie and Hellman [28] presented a key exchange protocol that allows the generation of a symmetric key based on the public key information. In a two-party setting, each party selects a secret key sk_1, sk_2. Using two publicly known numbers p and g, each participant computes and publishes their public key: $pk_i = g^{sk_i} \mod p$.

B

Table B.1: Notations for ECONoMy design.

Notation	Meaning
n	Number of participants in the protocol.
m	Number of items for which there is to be voted.
u	Number of possible classes the items can be classified as.
η	Number of zeros appended to plaintext encoding.
β	Number of bits required to represent one class in the vote encoding.
ξ	The bit size of the created masking value.
ψ	The size of an encrypted random number share.
λ	The size of a label.
$\mathscr{E}$	Denotes the use of AES encryption function.
$\mathscr{E}\prime$	Denotes the use of Paillier encryption function.
$\mathscr{D}$	Denotes the use of AES decryption function.
$\mathscr{D}\prime$	Denotes the use of the Paillier decryption function.
$\mathscr{I}$	Set of identifiers where $\mathscr{I} = \{1, 2, \dots, n\}$, where $\lvert\mathscr{I}\rvert = n$.
$\overset{\mathrm{r}}{\underset{x}{\leftarrow}}$	Uniformly selecting x random values out of a specific space, if omitted $x = 1$.
Π	Set of participants $\Pi = \{\pi_1, \pi_2, \dots, \pi_i\}$, where $\lvert\Pi\rvert = n$ and $i \in \mathscr{I}$
S	Unlabeled public samples, $\mathscr{S} = \{s_1, s_2, \dots, s_m\}$, where $\lvert S\rvert = m$.
$\mathscr{C}$	The possible classes, $\mathscr{C} = \{c_1, c_2, \dots, c_u\}$, where $\lvert\mathscr{C}\rvert = u$.
$\mathscr{L}$	Set of final labels, $\{L_1, L_2, \dots L_m\}$
Γ_{π_i}	Set of predictions for m items from participant i, $\{q_{\pi_i,s_1}, q_{\pi_i,s_2}, \dots, q_{\pi_i,s_m}\}$.
R_{π_i}	Denotes the original set of random numbers generated by participant π_i such that $R_{\pi_i} = \{r_{i1}, r_{i2}, \dots, r_{ij},\}$, for , $\pi_j \in \Pi$ and $i \neq j$.
R_{π_i,π_j}	Denotes the random shares generated by participant π_i, meant for participant π_j where π_i and $\pi_j \in \Pi$ and $i \neq j$.
R_{π_j,π_i}	Denotes the set of random numbers that participant π_i received from participant, π_j, generated using the random number generation. $R_{\pi_j,\pi_i} = \{r_{\pi_j,\pi_i}(1),$ $\mathrm{r}_{\pi_j,\pi_i}(2), \dots, r_{\pi_j,\pi_i}(m)\}$, where $\lvert R_{\pi_j,\pi_i}\rvert = m$
$\check{R}_{\pi_i,s_k}$	Denotes the set of random numbers of π_i, after aggregating all received part per item $1 \leq k \leq m$.
$\mathscr{M}_{\pi_i}$	Set of masked votes where $\mathscr{M}_{\pi_i} = \{\check{m}_{\pi_i,1}, \check{m}_{\pi_i,2}, \dots, \check{m}_{\pi_i,m}\}$ and $\check{m}_{\pi_i,k} = \check{R}_{\pi_i,s_k} + q_{\pi_i,k}$.
$\mathscr{M}_{s_k}$	Set of masked votes where $\mathscr{M}_{s_k} = \{\check{m}_{1,s_k}, \check{m}_{2,s_k},$ $\dots, \check{m}_{n,s_k}\}$ where $\check{m}_{k,s_k} = \check{R}_{\pi_k,s_k} + q_{\pi_i,k}$.
Δ	The set of all aggregated masked vote distributions.

Each party can now obtain their shared-secret by computing the following function:

$$key_1 = pk_2^{sk_1} = \left(g^{sk_2}\right)^{sk_1} ; key_2 = pk_1^{sk_2} = \left(g^{sk_1}\right)^{sk_2}$$

NOTATION
Table B.1 contains an overview of the used notations.

B.3. ECONoMy

In this section, we present our proposal, ECONoMy: a privacy-preserving collaborative ensemble protocol. We present the design of the ECONoMy protocol per phase, and each of the phases will show the required procedures and their respective analyses.

B.3.1. INITIAL SETUP

The first phase of the protocol initializes the required infrastructure and enables the participants to identify themselves by allowing them to participate in the protocol. Each individual $\pi_i \in \Pi$, in turn generates a key-pair $pk_i = g^{x_i}, sk_i = x_i$, corresponding to the Diffie-Hellman key-pairs. Additionally, every participant creates a Paillier [29] key pair, and all public keys are published. Each participant continues to generate shared AES keys in the following way. First, every participant $\pi_i \in \mathscr{P}$, publishes $n-1$ random values $\check{r}_{ij}$ encrypted using the public Paillier key of each receiving participant $\pi_j \in \Pi$ where $i \neq j$. Afterward, the shared AES-Key can be generated by each participant by computing the following function: $key = \left(pk_j\right)^{sk_i \cdot \check{r}_{ij} \cdot \check{r}_{ij}}$. resulting in a shared secret. This shared secret is then hashed to provide unique shared symmetric key of 128 bits suitable for AES-128. Finally, the participants agree on the vote encoding and a time frame within which all votes should be received.

At the end of this phase, the protocol is ready to commence, and every participant knows who is participating. All participant have access to an AES key for its communication with each participant in the protocol. Now we are ready to commence to the next phase, the generation of the masking values that will be used to hide each participant π_i's to be shared prediction values Γ_{π_i}.

B.3.2. RANDOM NUMBER GENERATION

We create random numbers using additive secret sharing, such that they sum up to zero, to mask each vote for all items in $\mathscr{I}$. By doing so, when all votes corresponding to an item $s_i \in \mathscr{S}$ are aggregated, the random values that masked the individual votes will be negated leaving us with the total of the vote distributions. To properly hide the underlying vote of bit size $(u \cdot \beta + \alpha)$, a random value of a specific bit-length would need to be chosen, corresponding to the currently required bit security, κ bits, according to NIST standards [30], where κ is our security parameter.

Every participant will need to generate m random numbers that each mask a prediction in Γ, as is done in the `GenerateRandomNumberShares` procedure in Algorithm 4. Therefore, the procedure first generates m random numbers that are ξ bits in length. Afterward, n parts are generated which modulo a prime p sum up to the original random value. While generating these parts, the computed shares are immediately ordered

Protocol 4 Random number generation protocol.

procedure GENERATERANDOMNUMBERSHARES(n, m)
  $R_{\pi_i} \xleftarrow[m]{r} \{0,1\}^{\xi}$
  $sharesPerRecipient \leftarrow \emptyset$
  $totals \leftarrow [\,]$
  for j in range(n), $j \neq i$ **do**
    $R_{\pi_i,\pi_j} \leftarrow \emptyset$
    for item in range(m) **do**
      $newShare \xleftarrow{r} \{0,1\}^{\xi}$
      $totals[item] += newShare$
      $R_{\pi_i,\pi_j}.include(newShare)$
    end for
    $sharesPerRecipient.include(R_{\pi_i,\pi_j})$
  end for
  $R_{\pi_i,\pi_i} \leftarrow \emptyset$
  for item in range(m) **do**
    $R_{\pi_i,\pi_i}.include((R_{\pi_i}[item]) - totals[item] \mod p)$
  end for
  $sharesPerRecipient.include(R_{\pi_i,\pi_i})$
  return $sharesPerRecipient$
end procedure

procedure ENCRYPTSHARES(R, PK_N)
  $encryptions \leftarrow \emptyset$
  $ctr = newCounter()$
  for x in range(n) **do**
    $encryptions.include(\mathscr{E}(R_{\pi_i,\pi_x}|ctr))$
  end for
  return encryptions
end procedure

procedure DECRYPTSHARES($\mathscr{E}(R), SK_i, R_{\pi_i}$)
  $R_{f,\pi_i} \leftarrow \emptyset$
  for e in range(m) **do**
    $r_{f,\pi_i}(e) \leftarrow 0$
    for d in range(n) **do**
      $r_{f,\pi_i}(e) += \mathscr{D}(\mathscr{E}(R|ctr)[d])$
    end for
    $r_{f,\pi_i}(e) -= R_a[p]$
    $R_{f,\pi_i}.include(r_{f,p_i}(e))$
  end for
  return R_{f,p_i}
end procedure

Protocol 5 Masking procedure, hiding the prediction values.

```
1: procedure MASKVOTES(R_{π_i,s_j}, Γ_{π_i})
2:     M_{π_i} ← ∅
3:     for j in range(m) do
4:         maskedValue ← R_{π_i,s_j} + Γ[i]
5:         M_{π_i}.insert(maskedValue)
6:     end for
7:     return maskedVotes
8: end procedure
```


according to their destination participant towards whom these particular shares are intended.

In order to transmit all shares in a single publishment, the shares are encrypted using the public key of the intended recipient using the `EncryptShares` procedure. A participant can, upon receiving all his or her parts, decrypt using the `DecryptShares` procedure, which aggregates the parts corresponding to the same item and subtracts her original random value. This operation leaves the participant with a final random value which can be used in the next phase, masking.

B.3.3. MASKING

We mask each vote to hide the contents of the vote provided by a participant. This masking can be done by simply adding the previously computed random value, to the corresponding vote value creating a masked value now containing the contents of the participant's vote (note that this can also be negative). Now, the resulting masked value can be shared with the other participants as can also be seen in Algorithm 5. Every participant $\pi_i \in \Pi$, where i is the participant identifier, executes this protocol for each item $i \in \mathscr{I}$. All masked votes can be shared in a single transaction, where a vote is contributed in concatenation with the hash of the original vote value. By doing so, it prevents the originating party to alter the original contributed value if he or she were to reveal the hidden information.

B.3.4. AGGREGATION

Now that the votes have been cast during the previous phase of the protocol, the totals can be computed. When adding all seemingly random-looking numbers, there will be a result that corresponds to the same value that would have been attained when all individual votes would have been aggregated without the use of the random numbers, as these values sum up to zero.

B.3.5. NOISE ADDITION AND FINAL MODEL GENERATION

Once the entire dataset is labeled, noise needs to be added according to the process described by Papernot et al. [31]. A participant proposes a noise distribution to extract perturbations from, which needs to be approved by the majority of other participants.

B.4. COMPLEXITY ANALYSIS

B.4.1. COMPUTATIONAL ANALYSIS

The computational complexity of ECONoMy is analyzed by reviewing the required number of operations per phase per party. The required operations are dependent on the number of participants n, as well as the number of items in the public data set m. The chosen size of the public dataset will depend on multiple factors, such as availability (i.e., what data can be acquired), as well as privacy requirements. The privacy obtained by transferring knowledge from the ensemble to the public data provides a bound to the number of queries done to the ensemble and limits privacy. The number of items in this public dataset resembles this bound and represents a trade-off between the amount of knowledge transferred that is available to train the global model on, and the level of privacy offered to the local training data. The number of participants is assumed to be smaller than the number of items to be labeled, even though this is not a constraint of the protocol. At least three parties should participate. When $n = 2$, the masking will be ineffective as a participant can retrieve the vote of its collaborator by detracting her own vote from the total.

Table B.2 shows the number of mathematical operations that occur in each phase.

Table B.2: Review of all operations occurring in a single execution of the ECONoMy protocol.

Phase	Operation	Total	Per Party
Initial Setup	Key Generations	n	1
Random Number Generation	Random number generation	$n \cdot m$	m
	Additive secret sharing	$n \cdot m$	m
	Encryptions/Decryptions	$2n^2 m$	$2nm$
	Additions	$m \cdot n$	m
Masking	Additions	$n \cdot m$	m
Aggregating	Additions	$n \cdot m$	$n \cdot m$
	Maximum	m	m

B.4.2. COMMUNICATION ANALYSIS

Information is transferred between participants in different phases of the ECONoMy protocol. The sizes of these messages are shown in Table B.3.It becomes apparent that the total amount of bits transferred in one execution of the protocol is equal to $mn^2\psi - mn\psi + nm\xi + m\lambda$.

B.5. EVALUATION

We present a prototype implementation for ECONoMy as well as comparison to *CrowdML*.

Table B.3: An overview of all communications needed for ECONoMy, and the size of the transferred information.

Phase	Operation	Total	Per party
Random Number Generation	Send	$mn^2\psi - mn\psi$	$mn\psi - m\psi$
	Retrieve	$mn^2\psi - mn\psi$	$mn\psi - m\psi$
Inference voting	Sharing votes	$nm\xi$	$m\xi$
Labeling	Publishing final labels	$m\lambda$	$m\lambda$

B.5.1. IMPLEMENTATION

The prototype implementation ECONoMy is written in Python 2.7 [32]. The accessibility of relevant packages such as *pycrypto* [33], provides the necessary cryptographic primitives, while scikit-learn [34], providing relevant machine learning models, making it suitable for our problem. We use the c4.4xlarge machine to execute the the prototype of ECONoMy, which offers 16 virtual vCPUs and 30 GiB memory [35].

B.5.2. RESULTS

The comparison performed between CrowdML and ECONoMy is made for varying values of participants (n) and to be predicted items (m). Table B.4 shows the run-time results obtained from these experiments.

As can be seen in Table B.4, the amount of time taken per vote appears to increase linearly by the number of included participants. Due to the significantly low time required per vote, the protocol can accommodate high values of both n and m. In CrowdML, the work-load is concentrated in a single entity, the central server. Figure B.2, shows the runtime if we were to divide the single-threaded experimental execution time of ECONoMy by the number of participants (n), displaying the work-load per party represented in time. CrowdML still outperforms ECONoMy for both $m = 500$ and $m = 1000$. At $n = 500$ ECONoMy takes approximately 5.4 times as long with an m of 500, while for $m = 1000$ this is reduced to 3.1 . These values show that there is an additional dependence on m for CrowdML that is not present in ECONoMy.

We continue to analyze this by looking at a per vote basis as depicted in Figure B.3. When we compare the experimental results for $m = 500$ and $m = 1000$, we see that the execution time is linear in the number of items to be predicted. In Figure B.3, we see both ECONoMy variants very close to one another, and an approximate doubling of execution time for CrowdML.

The experimental results show that the average time of a single prediction increases with increasing values of m. To corroborate this result, additional tests have been executed on each of the utilized classifiers separately, which shows that all appear to have this dependency on m.

B.6. PRIVACY ANALYSES

We assume a semi-honest adversarial model with an honest majority. The privacy of contributions provided by participants is argued using simulation-based security deduc-

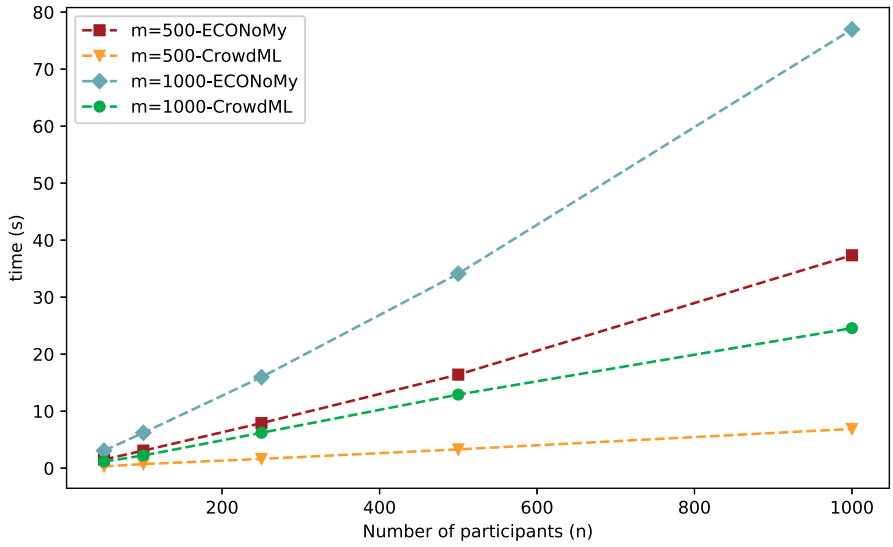

Figure B.2: Comparing the per participant computation time of ECONoMy with CrowdML.

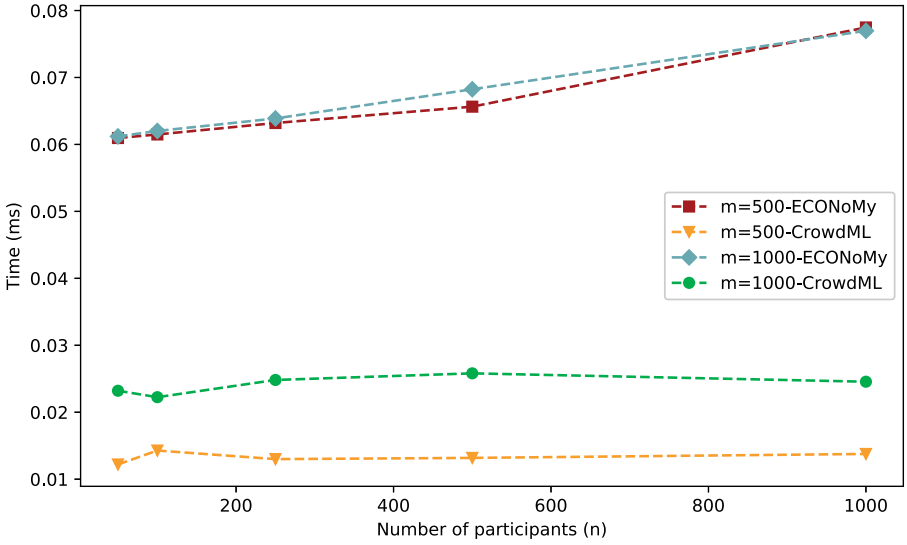

Figure B.3: Highlighting the dependency on m inherent to CrowdML, which is not present in ECONoMy.

Table B.4: The experimental results obtained by running both CrowdML and ECONoMy.

			ECONoMy			CrowdML	
n	m	nm	time (s)	$timen$	$(timen)vote$	time (s)	$timep$
50	500	25,000	76.193	1.524	6.096E-5	0.305	1.22E-5
	1000	50,000	153.008	3.060	6.120E-5	1.156	2.321E-5
100	500	50,000	307.432	3.074	6.149E-5	0.714	1.428E-5
	1000	100,000	620.036	6.200	6.201E-5	2.225	2.225E-5
250	500	125,000	1975.087	7.900	6.320E-5	1.619	1.30E-5
	1000	250,000	3991.550	15.966	6.388E-5	6.205	2.482E-5
500	500	250,000	8205.644	16.411	6.564E-5	3.296	1.318E-5
	1000	500,000	17,059.580	34.119	6.824E-5	12.907	2.581E-5
1000	500	500,000	37.345,950	37.346	7.747E-5	6.886	1.377E-5
	1000	1,000,000	76,970.465	76.970	7.697E-5	24.556	2.456E-5

tion [36].

- Let $f = (f_1, f_2, \dots f_n)$ be a functionality. We say that Φ securely computes f in the presence of static semi-honest adversaries, if an adversary cannot distinguish the protocol Φ form an ideal functionality f.
- The output of the protocol Φ on input $(\Gamma_{\pi_1}, R_{\pi_1}, \Gamma_{\pi_2}, R_{\pi_2}, \dots \Gamma_{\pi_n}, R_{\pi_n})$ and security parameter κ is denoted by $output^{\Phi}(\mathscr{M}'_{\Pi})$ and can be computed from all views of the execution.

We say Protocol Φ securely computes f in the presence of ρ semi-honest adversaries - simulators - $S_1, S_2, \dots S_\rho$ such that:

$$\begin{aligned} S_i(1^\kappa, \Gamma'_{S_i}, R'_{S_i}), f_i(\Gamma_{\pi_1}, R_{\pi_1}, \Gamma_{\pi_2}, R_{\pi_2}, \dots \Gamma_{\pi_n}, R_{\pi_n}) \stackrel{c}{\equiv} \\ view_i^{\Phi}((\Gamma_{\pi_1}, R_{\pi_1}, \Gamma_{\pi_2}, R_{\pi_2}, \dots \Gamma_{\pi_n}, R_{\pi_n}), \kappa) \end{aligned} \tag{B.4}$$

The communications of a single participant π_i with the other participants is visualized in Algorithm 6.

Let us assume an ideal functionality f that simulates a trusted third party and has access to perfect encryption and secure communication channels.

The simulated view can be represented as:

$$\begin{aligned} \{S_i(1^\kappa, \Gamma'_{S_i}, R'_{S_i}), f_i(\Gamma_{\pi_1}, R_{\pi_1}, \Gamma_{\pi_2}, R_{\pi_2}, \dots \Gamma_{\pi_n}, R_{\pi_n})\} \\ := \{(1^\kappa, \Gamma'_{S_i}, R'_{S_i}), \mathscr{M}'_{S_i}, \Delta')\} \end{aligned} \tag{B.5}$$

$$\begin{aligned} view_i^{\Phi}((\Gamma_{\pi_1}, R_{\pi_1}, \Gamma_{\pi_2}, R_{\pi_2}, \dots \Gamma_{\pi_n}, R_{\pi_n}), \kappa) \\ := \{(1^\kappa, \Gamma_{\pi_i}, R_{\pi_i}), \mathscr{M}_{\pi_i}, \Delta)\} \end{aligned} \tag{B.6}$$

For any distinguisher D and negligible function $\alpha(\cdot)$:

$$\begin{aligned} |Pr[D(1^\kappa, \Gamma'_{S_i}, R'_{S_i}), \mathscr{M}'_{S_i}, \Delta') = 1] - \\ Pr[D(1^\kappa, \Gamma_{\pi_i}, R_{\pi_i}), \mathscr{M}_{\pi_i}, \Delta) = 1]| \leq \alpha(\kappa)\,. \end{aligned} \tag{B.7}$$

Protocol 6 Communication of a participant π_i with the protocol Φ

π_i		Φ
$\texttt{view}^{\Phi}_{\pi_i}(\Gamma_{\pi_i}, R_{\pi_i})$		$\texttt{view}^{\Phi}(\mathscr{M}_{\pi_i})$
$R = \{R_{\pi_i,\pi_j} \mid j \neq i\}$	$\xrightarrow{R}$	
	$\xleftarrow{R_J}$	$R_J = \{R_{\pi_j,\pi_i} \mid j \neq i\}$
$\breve{R}_{\pi_i,s_k} = \sum_{j=0}^{n} R_{\pi_j,\pi_i}(k) - r_{ik}$ $\breve{R}_{\pi_i} = \{\breve{R}_{\pi_i,s} \mid s \in S\}$ $\mathscr{M}_{\pi_i} = \Gamma_{\pi_i} + \breve{R}_{\pi_i}$	$\xrightarrow{\mathscr{M}_{\pi_i}}$	
		$\mathscr{M} = \{\mathscr{M}_s \mid s \in S\}$ $\Delta_s = \sum_{i=0}^{n} \breve{m}_{i,s}$
	$\xleftarrow{\mathscr{M},\Delta}$	$\Delta = \{\Delta_s \mid s \in S\}$

An adversary controlling a minority ρ nodes, has access to a subset of the shared information. The random shares in R_j are encrypted using the assumed semantically secure AES. The provided confidentiality causes an adversary only to observe the shares aimed for or originating from the ρ participants. Every masking value is dependent on one random share from each participant, and the original random value generated by the victim. This dependency can be shown as follows:

$$\breve{R}_{\pi_v,s_k} = \sum_{j=0}^{\rho} R_{\pi_j,\pi_v}(k) + \sum_{l=\rho+1}^{n} R_{\pi_l,\pi_v}(k) - r_{vk} \tag{B.8}$$

for victim π_v. The adversaries are unable to reconstruct r_{vk}, as this value is split into n parts from which the adversary is only able to access ρ.

B.7. CONCLUSION

This paper presents a light-weight and efficient protocol called ECONoMy, which utilizes masking technique to preserve participants' privacy within a decentralized ensemble based collaborative learning setting. Our protocol is proposed under the semi-honest security model, and is suitable in settings with large participants such as with IoT devices.

Protocol 7 Communication of simulator S_i with the ideal functionality f

S_i		f
$\texttt{view}^f_{S_i}(\Gamma'_{S_i}, R'_{S_i})$		$\texttt{view}^f(\mathscr{M}'_{S_i})$
$R' = \{R'_{S_i,\pi_j} \mid j \neq i\}$	$\xrightarrow{R'}$	
	$\xleftarrow{R'_J}$	$R'_J = \{R'_{\pi_j,S_i} \mid j \neq i\}$
$\check{R}'_{S_i,s_k} = \sum_{j=0}^{n} R'_{\pi_j,S_i}(k) - r'_{ik}$ $\check{R}'_{S_i} = \{\check{R}_{S_i,s} \mid s \in S\}$ $\mathscr{M}'_{S_i} = \Gamma'_{S_i} + \check{R}'_{\pi_i}$	$\xrightarrow{\mathscr{M}'_{S_i}}$	
		$\mathscr{M}' = \{\mathscr{M}'_s \mid s \in S\}$ $\Delta'_s = \sum_{i=0}^{n} \check{m}'_{i,s}$ $\Delta' = \{\Delta'_s \mid s \in S\}$
	$\xleftarrow{\mathscr{M}',\Delta'}$	

REFERENCES

[1] D. S. Terzi, R. Terzi, and S. Sagiroglu, *A survey on security and privacy issues in big data,* in *10th International Conference for Internet Technology and Secured Transactions, ICITST 2015, London, United Kingdom, December 14-16, 2015* (2015) pp. 202–207.

[2] S. Sagiroglu and D. Sinanc, *Big data: A review,* in *2013 International Conference on Collaboration Technologies and Systems, CTS 2013, San Diego, CA, USA, May 20-24, 2013* (2013) pp. 42–47.

[3] N. Papernot, P. D. McDaniel, A. Sinha, and M. P. Wellman, *Sok: Security and privacy in machine learning,* in *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018* (2018) pp. 399–414.

[4] Louis Columbus, *10 ways machine learning is revolutionizing manufacturing in 2018,* `https://www.forbes.com/sites/louiscolumbus/2018/03/11/10-ways-machine-learning-is-revolutionizing-manufacturing-in-2018/#66554f0423ac` Online; accessed April, 2018.

[5] M. Chui, *Artificial intelligence the next digital frontier?* McKinsey and Company Global Institute , 47 (2017).

[6] I. Kononenko, *Machine learning for medical diagnosis: history, state of the art and perspective,* Artificial Intelligence in Medicine **23**, 89 (2001).

[7] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, *End to end learning for self-driving cars,* CoRR **abs/1604.07316** (2016), arXiv:1604.07316 .

[8] I. H. Witten, F. Eibe, and M. A. Hall, *Data mining: practical machine learning tools and techniques, 3rd Edition* (Morgan Kaufmann, Elsevier, 2011).

[9] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, fourth edition* (Elsevier, 2009).

[10] P. Laskov and R. Lippmann, *Machine learning in adversarial environments,* Machine Learning **81**, 115 (2010).

[11] O. Dictionary, *Definition of the word 'adversary'.* (2018).

[12] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar, *Adversarial machine learning,* in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, AISec 2011, Chicago, IL, USA, October 21, 2011* (2011) pp. 43–58.

[13] J. Hamm, A. C. Champion, G. Chen, M. Belkin, and D. Xuan, *Crowd-ml: A privacy-preserving learning framework for a crowd of smart devices,* in *35th IEEE International Conference on Distributed Computing Systems, ICDCS 2015, Columbus, OH, USA, June 29 - July 2, 2015* (2015) pp. 11–20.

[14] P. Sollich and A. Krogh, *Learning with ensembles: How overfitting can be useful,* in *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, USA, November 27-30, 1995* (1995) pp. 190–196.

[15] L. Rokach, *Ensemble-based classifiers,* Artif. Intell. Rev. **33**, 1 (2010).

[16] R. Shokri and V. Shmatikov, *Privacy-preserving deep learning,* in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015* (2015) pp. 1310–1321.

[17] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, *Deep models under the GAN: information leakage from collaborative deep learning,* in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017* (2017) pp. 603–618.

[18] J. Hamm, Y. Cao, and M. Belkin, *Learning privately from multiparty data,* in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016* (2016) pp. 555–563.

[19] A. Kurakin, I. J. Goodfellow, and S. Bengio, *Adversarial examples in the physical world,* CoRR **abs/1607.02533** (2016), arXiv:1607.02533 .

[20] D. Meng and H. Chen, *Magnet: A two-pronged defense against adversarial examples,* in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017* (2017) pp. 135–147.

[21] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, *Universal adversarial perturbations,* in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017* (2017) pp. 86–94.

[22] J. Su, D. V. Vargas, and K. Sakurai, *One pixel attack for fooling deep neural networks,* CoRR **abs/1710.08864** (2017), arXiv:1710.08864 .

[23] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, *Membership inference attacks against machine learning models,* in *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017* (2017) pp. 3–18.

[24] M. Fredrikson, S. Jha, and T. Ristenpart, *Model inversion attacks that exploit confidence information and basic countermeasures,* in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015* (2015) pp. 1322–1333.

[25] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, *Stealing machine learning models via prediction apis.* in *USENIX Security Symposium* (2016) pp. 601–618.

[26] K. Kursawe, G. Danezis, and M. Kohlweiss, *Privacy-friendly aggregation for the smart-grid,* in *Privacy Enhancing Technologies - 11th International Symposium, PETS 2011, Waterloo, ON, Canada, July 27-29, 2011. Proceedings* (2011) pp. 175–191.

[27] F. D. Garcia and B. Jacobs, *Privacy-friendly energy-metering via homomorphic encryption,* in *Security and Trust Management - 6th International Workshop, STM 2010, Athens, Greece, September 23-24, 2010, Revised Selected Papers* (2010) pp. 226–238.

[28] W. Diffie and M. E. Hellman, *New directions in cryptography,* IEEE Trans. Information Theory **22**, 644 (1976).

[29] P. Paillier, *Public-key cryptosystems based on composite degree residuosity classes,* in *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding* (1999) pp. 223–238.

[30] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, *Recommendation for key management part 1: General (revision 3),* NIST special publication **800**, 1 (2012).

[31] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, *The limitations of deep learning in adversarial settings,* in *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016* (2016) pp. 372–387.

[32] P. S. Foundation., *Python 2.7,* (2010).

[33] D. C. Litzenberger, *Python cryptography toolkit (pycrypto),* (2018).

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python,* Journal of Machine Learning Research **12**, 2825 (2011).

[35] Amazon, *Amazon ec2 c5 instances,* (2018).

[36] Y. Lindell, *How to simulate it - A tutorial on the simulation proof technique,* in *Tutorials on the Foundations of Cryptography.* (2017) pp. 277–346.

ACKNOWLEDGEMENTS

Pursuing a Ph.D programme use to be a wild dream of mine, but some people have contributed directly and indirectly to making it both a reality and a success. This is where I appreciate some of those people who have believed in me enough to nudge me over that finish line, in one way or another. This journey was not always rosy and linear, especially when I had to be thousands of kilometers from by home country, but the people I met spiced it enough to make me not give up.

Firstly, I want to appreciate the best daily supervisor there is, Zeki Erkin. Our journey started with that simple email of mine asking for an opportunity and you were willing to give me a chance. I will forever be grateful. I recall meeting you for the first time on February 18, 2016 and asking a selfie, that is how excited I was to start my Ph.D journey with you. Your comments during our technical meetings and feedback on my first article's draft are still fresh in my memory and I have become a better researcher under your tutelage. Thank you Zeki, you deserve a chocolate cake.

To my promotor Prof. Inald Lagendijk, you made my experience in TU Delft worth every moment. You were always willing to provide advise, support, guidance and technical comments to my ideas and works. I am aware that I probably stressed you more than most Ph.D students you have come across, but best believe that I squeezed out every iota of lesson there was to learn from interacting with you. You made me continue my studies with the ease and knowledge that I had a family away from home.

My appreciation goes to my committee members, Marcel Reinders, Cees Vuik, Michel Dumontier, Mauro Conti and Stefan Katzenbeisser for making out time to review my thesis, providing constructive feedback and participating in my defense ceremony. Also to Jeroen Slobbe and Colin Schappin for aiding me in translating the summary and propositions of this thesis. To those who helped proofread some part of my work, I say thank you to Echeta Zal, Nneka Uguwoke, Ijeoma Ositadimma, Lois Ubani, Uchenna Ezeora and Oma Ajanwachukwu.

The daily life of a Ph.D student includes those with whom you have a brainstorming session, coffee meetings and chit chats, and I had a fair share of those. To my favourite colleague Gamze, you are special and a dependable sister. I am grateful to Oguzhan for all the memories and technical meetings, your cake recipe stands out. Thanks to Majid for all the support and brainstorming sessions. Also my appreciation to the queen of the group Sandra, and Saskia your contributions ware enormous. Other people who deserve appreciation include Harm, Vincent, Azqa, Manel, Jehan, Bartosz, Stjepan, Sicco and Miray.

Back in Nigeria, there were people whose support were immeasurable. To my aunt Dorathy Ezeora and Prof. and Dr. Mrs T.M Okonkwo for being all that a parent should be for me and signing off on my first journey to London, I sincerely appreciate. To my friend and brother Richard Ugwuanyi for all the support during the difficult years of a Ph.D

programme, I remember you and would always have your back. Thanks to Chibueze Agu and Emmanuel Ugwuonna for the encouragement and advise.

To my friends Chibuzo Okolo, Iroegbu Iroegbu, Austin Ogwo, Chuka Okeke, Arinze Ifedobi, Precious Aguecheta, Oluchi Oliobi, Sam Ode and Chinedu Ugwuanyi, keeping in touch meant a whole lot to me.

My appreciation goes to the memories of two special people I lost during this journey: to my good friend, late Ray Obiora Okonkwo, I miss you and will always remember the dreams we had. And to my grandmother late Mama Nkechi, may your soul rest in peace.

To my parents, I am indebted to you for your contribution to my education, it provided me the foundation upon which I could dream.

Lastly, I am grateful to my wife Nneka Ugwuoke and kids, Adanna and Mmasinachi for always providing me a reason to hold on and give my best. This journey would not have been possible without you.

CURRICULUM VITÆ

Chibuike Innocent UGWUOKE

Chibuike Ugwuoke was born in Nsukka, Nigeria on December 12, 1987. He graduated high High School from Konigin Des Friedens College Enugu, in 2005. In 2011, he obtained his B.Sc in Computer Science from Computer Science Department in University of Nigeria, Nsukka. Chibuike participated in the 2012 Presidential Special Scholarship Scheme for Innovation and Development (PRESSID), initated by the President Goodluck Jonathan administration. He was ranked 48th out of the more than 2000 first class graduates that applid for the scheme. Following the award of PRESSID, in 2015, Chibuike obtained his M.Sc in Information Security from the University College, London.

While in London, Chibuike first got introduced to cryptography and was curious and excited to take on his M.Sc dissertation in the area of Private Information Retrieval (PIR). He subsequently pursued a Ph.D programme in the field applied cryptography in Delft University of Technology, under the supervision of Prof. dr. ir R.L Lagendijk and dr. Zekeriya Erkin, where he investigated genome privacy from a cryptographic perspective. During his Ph.D, Chibuike represented Delft University of Technology in the annual iDASH challenge of 2018 and 2019, and his team's contribution made it to the top 6 solutions in both years. Chibuike also supervised two master students on their dissertation and further supervised three bachelor students on their mid-session group project. He also contributed to the Cryptography course as a teaching assistant.

As of May 2020, Chibuike joined the Cryptography team at Deloitte Netherlands as a senior consultant.