Alberto BLANCO ÁLVAREZ

Master's Thesis

Privacy-preserving Sharing of Genomic Data

Directed by Dr. Josep DOMINGO FERRER

Computer Security Engineering and Artificial Intelligence Degree

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2022

Acknowledgements

I would like to thank Josep Domingo Ferrer for agreeing to be my tutor for the realization of this project, for having introduced and taught the essentials of privacy protection of data, both as a teacher and as the guide for this project, and for having helped me during the process of elaborating this work, as without his help, it would have been completely different developing and investigating the world of privacy security in the biotechnological aspect, a field still in development, hard to investigate without help.

I would also like to thank, with the bottom of my hearth, my parents and my brother, who have supported me through my whole life, student or not, and have allowed me to get to the place where I am at the moment, aiding me whenever I needed it, unconditionally. I would not be the men I am right now without them by my side, and thus, I would love to express my deepest gratitude to them. For everything.

Contents

List of Figures **8**

Abstract **9**

1 Introduction **11**

1.1 Motivation 11

1.2 Our proposal 12

1.3 Organization of the document 13

1.4 Glossary 14

2 Preliminaries **16**

2.1 Introduction 16

2.1.1 Genomic data 17

2.1.2 High-throughput methods 19

2.1.3 Storing genomic sequence data 22

2.1.4 Privacy in genomic data 24

2.1.5 Privacy problems to be solved 27

2.2 Privacy preserving techniques 28

2.2.1 K-anonymity 28

2.2.2 Homomorphic encryption: 28

2.2.3 CKKS 30

2.2.4 RNS-CKKS 31

2.2.5 ResNet 32

2.2.6 Pseudorandom function 33

2.2.7 Message Authentication Codes 34

2.2.8 Oblivious Transfer 35

2.2.9 Garbled circuit 36

2.2.10 METIS system 38

3 A computation over encrypted data in cloud environments for genomic data **44**

3.1 Overview 44

3.2 Description of the protocol 44

3.2.1 Improvements on the METIS scheme 46

4 Implementation of the synthetic genomic data generation algorithm **53**

4.1 Overview 53

4.2 Description of the algorithm 54

4.3 Analysis of the results 58
4.3.1 Haploview software 58
4.3.2 Algorithm results 62
5 Conclusions and future work 72
5.1 Future work 73
6 Bibliography 74

List of Figures

Figure 1: Simplified structure of the DNA. 18
Figure 2: SNP in genetic data. 18
Figure 3: Steps of high-throughput assays in genome biology. 20
Figure 4: FASTQ file format. 23
Figure 5: Commonly used Phred quality scores in different sequencing techniques. 23
Figure 6: Recombination and crossover schemes of genetic material. 25
Figure 7: Genotype and haplotype example. 25
Figure 8: Allele frequencies preservation in synthetic genomic microdata generation. 26
Figure 9: Crossover and mutation events simulation. 26
Figure 10: High level view of CKKS. 30
Figure 11: The specification of ResNet-20 (CIFAR-10) 32
Figure 12: Canonical form of a residual neural network. A layer $\ell - 1$ is skipped over activation from $\ell - 2$. 33
Figure 13: 1-out-of-2 OT protocol (Schneier, 1996) 36
Figure 14: Example of the construction of the garbled table 37
Figure 15: Structure of the METIS system 39
Figure 16: METIS algorithm 43
Figure 17: Comparison of two-round $OTnk$ in terms of communication cost. 48
Figure 18: Prefix tree generation states. 50
Figure 19: Example of fully constructed prefix tree. 51
Figure 20: Example query for a prefix tree. 52
Figure 21: Reference of .ped file 55
Figure 22: Reference of .info file. 55
Figure 23: Monocartenary DNA assimilation process. 56
Figure 24: Base substitution mutations: Transitions and transversions. 57
Figure 25: Data anonymization iterating by desired amount of generations 58
Figure 26: Marker information checks in Haploview. 60
Figure 27: Haploview LD display with recombination rate plotted. 61
Figure 28: Haploview Haplotype display with transition rates and pattern matches rate. 61
Figure 29: Chron disease patients data 62
Figure 30: Marker information for Chron patients 63
Figure 31: LD display for Chron patients 63
Figure 32: Haplotype display for Chron patients 64

Abstract

The technological advancements that science has achieved in the latest years in the field of medicine allowed the creation of new fields of study that researched new medical approaches that aimed to personalize therapies, tailoring strategies based on the genomic, epigenomic and proteomic profiles of an individual to provide the most effective treatment for an individual or recognizing hidden abnormalities before the disease is manifested. This is due to the improvements in genomic sequencing techniques, which provide more insightful data for researchers while diminishing the costs of said investigations.

While the uprising in the development of improved sequencing methods and personalized medicine are to be highly insightful for new techniques, they demand large quantities of data of many individuals to be processed to aid the research, data that is highly sensitive of the information of an individual, yet few specific regulatory standards protect the security of the sensitive health data needed for precision medicine research.

Genomic data are extremely sensitive personal data, as they contain information that is unique to an individual and very disclosive regarding their health, expected diseases, skills, etc. At the same time, disclosing an individual's genomic data also substantially discloses the genomic information of their kin. Anonymizing genomic data to make them non-personal and hence shareable under the General Data Protection Regulation (GDPR) is challenging, as the usual anonymization methods based on data perturbation or generalization might have unknown consequences, because the meaning of such changes on genomic data is not yet well understood. What exists is outdated and insufficient, thus new methods to protect the anonymity of the data are to be investigated so as not to endanger it being compromised by cyberattacks.

In this thesis, we explore the state of the art in privacy-preserving methods for genomic data, and describe a new, detailed cryptographic protocol for privacy preservation in genomic data sharing that could guarantee the security of the data while not hampering the progress of new research.

Chapter 1

1 Introduction

1.1 Motivation

A new systems approach to diseased states and wellness result in a new branch in the healthcare services, namely, personalized medicine (PM) [34]. To achieve the implementation of PM concept into the daily practice including clinical cardiology, it is necessary to create a fundamentally new strategy based upon the subclinical recognition of bioindicators (biopredictors and biomarkers) of hidden abnormalities long before the disease clinically manifests itself.

In recent years, technological improvements have substantially decreased the cost of genome sequencing, thus generating an unprecedented amount of genomic data that have been vital in many research applications. Several research initiatives (for example, the National Institutes of Health (NIH) All of Us Research Program) are integrating these data with the goal of serving as a source of analyses for a wide range of studies. Simultaneously, genomic-data-driven applications in the private sector have substantially expanded, wherein personal genomic data are collected to provide health-related services to individuals.

In parallel with this, there has been an uprising in the investigation on personalized medicine, a particularly novel and exciting topic in the medicine and healthcare industries, that has the potential to transform medical interventions by providing effective, tailored therapeutic strategies based on the genomic, epigenomic and proteomic profile of an individual, whilst also remaining mindful of a patient's personal situation, by tailoring a treatment as individualized as the disease. The approach relies on identifying genetic, epigenomic, and clinical information that allows the breakthroughs in our understanding of how a person's unique genomic portfolio makes them vulnerable to certain diseases [23], [24], [34].

But this precision has a price that science and medicine do not acknowledge. Personalized medicine demands that we all contribute our medical histories and genomes to the big data research pool. The science works only if the numbers are exceptionally large; so large that some envision every patient as a subject whose medical data will be shared for research, while assuring the privacy of the individual.

A growing number of experts, particularly re-identification scientists, believe it is simply not possible to de-identify the genomic data and medical information needed for precision medicine. To be useful, such information cannot be modified or stripped of identifiers to the point where there's no real risk that the data could be linked back to a patient [35].

Security is essential to protect privacy, yet few specific regulatory standards protect the security of the sensitive health data needed for precision medicine research. What exists is outdated and insufficient.

For example, while a fingerprint, long known to be useful for re-identification, was protected under the 2003 federal HIPAA Privacy and Security rules as a "biometric identifier," the genome is not—and today, 14 years and millions of genotypes later, still is not.

Growing number of civilians have seen their medical data compromised by cyberattacks on data stored at hospitals, insurers, and clinical laboratories, mainly attacking electronic health records (EHRs), which link to the genomic data needed for precision medicine. If said data were to be obtained, the privacy impact of cyberattacks will increase exponentially, as, unlike a medical record number or credit card number, genome sequences can't be replaced when compromised, and sequence data are a wellspring of information about health risks, ancestry, and sometimes, unexpected parenthood [36].

Databases containing genomes and medical histories are multiplying, sometimes populated with the data of unwitting participants who don't know researchers have sequenced their genomes and placed the data in research databases operated by public entities (such as the NIH) or private drug companies. Data from these databases is shared with researchers world-wide, typically under a "data use agreement" that offers no recourse to data subjects if their information is misused or compromised.

Thus, the main object of this work is to investigate the current state of art in the field of privacy protection technologies applied to genomics, and to build a system that can guarantee the security of our genetic data while allowing working with the data for new investigations without hampering its progress and avoiding re-identification attacks.

1.2 Our proposal

We propose a protocol based on the METIS protocol, which promotes the division of the system into four different parties that each have their own roles, communicating researchers and data owners as a service provider, while encrypting the genomic data using model methods that artificially compute future generations' data in order to grant anonymity.

In general, there are three main problems that exists in working with encrypted genomic data, which are securing against reidentification attacks, secure computing and storing the genomic data, and query privacy. Current methods fail to provide defense against these three problems, are not yet investigated, or work only in the theoretical aspect, not

considering the properties of genomic data (such as needing long sequences of data to be of any use, instead of using small-scale data).

Our method relies on the properties of garbled circuits, which allow the computation of functions on the data without revealing any information, while easily allowing the data owners to allow or reject any clients by sending them the key that allows to correctly evaluate their function. This way, the data owners have complete control over the genomic data.

The clients will need to be authenticated and certified, via a legal contract, following similar systems in genomic databases, such as the eICU collaborative Research Database, where our system acts as a service provider system between genetic researchers and data owners, providing computations on large sets of genetic samples in a remote, secure way and diminishing the storage requisites of the data owners on genomic data.

We also implement improvements over existing genomic security schemes as implementing modern Oblivious Transfer schemes that solve efficiency issues that surged on existing systems in the biological field, and the possibility to execute secure count queries on the encrypted data to allow the investigation on SNPs employing a prefix tree system.

We add security against reidentification attacks by not using the original sequenced data, but instead, future generations' artificially computed data, which by randomly pairing individuals, we can allow the statistical data of the alleles to remain identical while protecting current and future individuals' data. For this, we propose an algorithm that takes advantage of the properties of the phenotypic proportions that apply to the genome sequences, supporting the privacy of the genomic data before it's sent from the sequencing center.

We modify the obtained genomic by simulating the recombination events in the mitosis, recombining all the individual's data as to produce new sequences that can be used for investigation.

1.3 Organization of the document

The rest of the document is structured as follows.

In the second chapter we will introduce the biological concepts needed to understand in more detail genomic data, what it is, how it can be obtained and used, and more specially, how can we use it in the technological sector. We will describe DNA, SNPs, genomic sequencing techniques and the current state of technology used in the production of such data. Then it will introduce the problem that this study aims to solve and the different techniques that are currently used in the privacy protection oriented to genomic data sector.

The following chapter will be devoted to describing the systems and schemes on which our solution is based upon, mainly centering in the METIS system.

Next, we will describe our proposed system, providing a general overview of it, the different actors into which the system is decomposed, the technology necessary to implement it and real propositions of applications, as well as introducing our algorithm to aid in the anonymization of genomic databases through artificial generation of new generations, as well as studying its results with the aid of the haplotype analysis software '*Haploview*'.

Finally, we will provide a conclusion and future plans for our work.

1.4 Glossary

DNA: Desoxyribonucleic acid, a thread-like chain of nucleotides molecules carrying the genetic instructions used in the growth, development, functioning and reproduction of all known living organisms and many viruses.

RNA: Ribonucleic acid, a polymeric molecule assembled as a chain of nucleotides in one single strand that participates in the biological roles of coding, decoding, regulation and expression of genes.

Nucleotide: Organic molecules consisting of a nucleoside and a phosphate. They serve as monomeric units of the DNA and the RNA.

Meiosis: One of the types of cell division of germ cells in sexually reproducing organisms that produces the gametes. It involves two rounds of division that ultimately result in four cells with only one copy of each chromosome (haploid cells).

Chromosome: Long DNA molecule with part or all of the genetic material of an organism. They are condensed during the initial phases of cell division, where they display a complex three-dimensional structure, which plays a significant role in transcriptional regulation.

Chromatid: One of the halves of a duplicated chromosome. In replication, the DNA molecule is copied, and the two molecules are known as chromatids. During the later stages of cell division these chromatids separate longitudinally to become individual chromosomes.

Chromosomal crossover: One of the final phases of genetic recombination, where an exchange of genetic material occurs during sexual reproduction between two homologous chromosomes' non-sister chromatids that results in recombinant chromosomes.

Gene: Basic unit of heredity, sequence of nucleotides in DNA that are transcribed to produce a functional RNA.

Allele: A variation of the same sequence of nucleotides at the same place on a long DNA molecule. Its simplest form are SNPs.

SNP: Single-nucleotide polymorphism, a substitution of a single nucleotide at a specific position in the genome that needs to be present in a sufficiently large fraction of the population (e.g., 1% or more).

Heterozygosity: The percentage of gene loci that are heterozygous in an average individual of a given population.

Gene locus: A locus refers to a fixed position in a chromosome, thus allowing to determine the position of a gene or marker in a chromosome.

Genotype: The complete set of genetic material for an organism. It refers to the alleles or variants an individual carries in a particular gene or genetic location. In diploid species like humans (which carry two copies of each chromosome), two full sets of chromosomes are present, meaning each individual has two alleles for any given gene.

Haplotype: Group of alleles in an organism that are inherited together from a single parent. The haploid genotype (haplotype) is a genotype that considers the singular chromosomes rather than the pairs of chromosomes. It can be all the chromosomes from one of the parents or a minor part of a chromosome, such a smaller sequence of base pairs.

Genome: Nucleotide sequences of DNA (or RNA in RNA viruses) that conforms all the genetic information of an organism, including protein-coding genes and non-coding genes, the other functional regions of the genome (Non-coding DNA), and any junk DNA if present.

Polymorphism: It refers to the occurrence of two or more clearly different morphs, phenotypes, in the population of a species

High-throughput methods: Drug discovery process that allows automated testing of large numbers of chemical and/or biological compounds for a specific biological target. It aims to identify through compound library screenings, candidates that affect the target in the desired way.

DNA sequencing: The process of determining the nucleotide order of a given DNA fragment. Sequencing results in a symbolic linear depiction known as a sequence which succinctly summarizes much of the atomic-level structure of the sequenced molecule.

Personalized medicine: Also referred as precision medicine, it is a medical model that separates people into diverse groups—with medical decisions, practices, interventions and/or products being tailored to the individual patient based on their predicted response or risk of disease.

FASTQ: Text-based format for storing both a biological sequence (usually nucleotide sequence) and its corresponding quality scores. Both the sequence letter and quality score are each encoded with a single ASCII character for brevity, used to store a variable number of sequence records for genetic material.

Chapter 2

2 Preliminaries

2.1 Introduction

Personalized medicine promises much. New initiatives aim to harness technology and genomics to create bespoke medicine, customizing your healthcare like your Facebook profile. Instead of relying on generic practice guidelines, your doctors may one day use these new analytic tools to find the ideal treatment for you. Big data will make this precision possible: patterns that emerge from the DNA and medical records of millions can predict which treatments work best for which patients. Fewer mistakes, lower costs, and more effective care may result.

An individual's genetic information is possibly the most valuable personal information. While knowledge of a person's DNA sequence can facilitate the diagnosis of several heritable diseases and allow personalized treatment, its exposure comes with significant threats to the patient's privacy.

Several notable features make genomic data different from other health data. For example, genomic data carry information that can be effectively used for determining the prognosis of health conditions (for example, Alzheimer's disease) and designing preventive interventions. Another important property of genomic data is the presence of significant commonality among individuals who are blood relatives. Therefore, genome analysis is commonly used for susceptibly risk, paternity, and relativeness testing (for example, ancestry services), and for forensic purposes (for example, genomic genealogy searches).

The development of personalized medicine is tightly linked with the correct exploitation of molecular data, which derives to an increasing demand to share these data for research purposes. Transition of clinical data to research is based in the anonymization of these data so the patient cannot be identified, the use of genomic data poses a great challenge because its nature of identifying data [23].

To support large-scale biomedical research projects, organizations need to share person-specific genomic sequences without violating the privacy of their data subjects. In the past, organizations protected subjects' identities by removing identifiers, such as name and social security number; however, recent investigations illustrate that deidentified genomic data can be "reidentified" to named individuals using simple automated methods.

This motivates the demand for cryptographic protocols which enable computation over encrypted genomic data while keeping the owner of the genome in full control, as traditional privacy models designed for health data provide limited protection of genomic data. An attacker may learn sensitive information about a target individual by exploiting the dependency between genomic data and other publicly available information such as family name, demographic data and observable features (for example, eye and hair color).

Because personal data are readily available (for example, through social networks), privacy assurances through traditional methods are unlikely to be sustainable.

Furthermore, the rise of direct-to-consumer (DTC) genetic testing companies poses new privacy and ethical concerns. These companies are collecting data from a growing number of individuals, some of whom may share their data without fully understanding the potential implications for themselves as well as their current and future blood relatives.

Privacy breaches can have serious social implications and can adversely affect genomic-driven research, such as by decreasing data collection and data sharing. Therefore, it is imperative to ensure privacy both as a fundamental right for individuals and as an enabling strategy to support responsible data sharing.

With this objective, this study aims at exploring the state of the art in privacy-preserving methods for genomic data, familiarizing with the current privacy-preserving techniques that preserve genomic utility, and at proposing new methods that preserve the utility of genomic data while encrypted.

2.1.1 Genomic data

Genomic data refers to the genome and DNA data of an organism. They are used in bioinformatics for collecting, storing, and processing the genomes of living things. Genomic data generally require a large amount of storage and purpose-built software to analyze.

The **DNA**, or deoxyribonucleic acid, is a polymer composed of two polynucleotide chains that coil around each other to form a double helix carrying genetic instructions for the development, functioning, growth and reproduction of all known organisms and many viruses.

In humans, the genetic material is formed by two strands of DNA joined together by the nitrogenous bases, linking adenine with thymine and guanine with cytosine.

It consists of the hereditary material in almost all organisms, found in the nucleus of the cells (generally) and in the mitochondria. It stores the genetic information of an individual where each strand of the DNA is known as a **polynucleotide**, composed of simpler monomeric units, the **nucleotides**. These nucleotides are composed on a nitrogen-containing nucleobases, the nitrogenous bases, which conform a code based on four different chemical bases: adenine (A), guanine (G), cytosine (C), and thymine (T). These nucleotides are jointed to one another in a chain by covalent bonds between the sugar of one nucleotide and the phosphate of the next, allowing both chains to be linked.

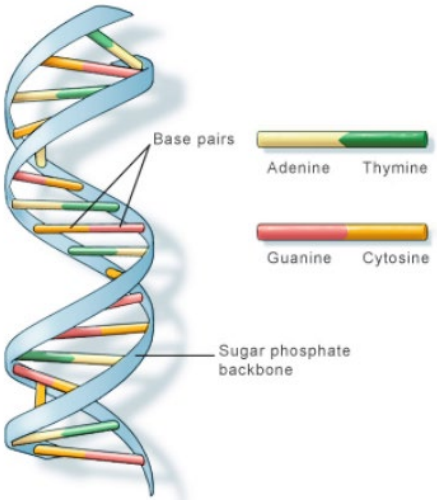


Figure 1: Simplified structure of the DNA.

Human DNA consists of about 3 billion bases, and more than 99 percent of those bases are the same in all people. The order, or sequence, of these bases determines the information available for building and maintaining an organism, similar to the way in which letters of the alphabet appear in a certain order to form words and sentences.

The DNA forms a self-replicating structure, known as **chromosomes**, which are the units that carry genetic information. Human cells have 23 pairs of chromosomes (22 pairs of autosomes and one pair of sex chromosomes), giving a total of 46 per cell.

The main object of study in genetics are the precise positions along a chromosome where the DNA of different people may vary. These positions are named **Single Nucleotide Polymorphisms (SNP)**. A SNP is a single base pair change in the DNA sequence at a particular point compared with the "common" or "wild-type" sequence.

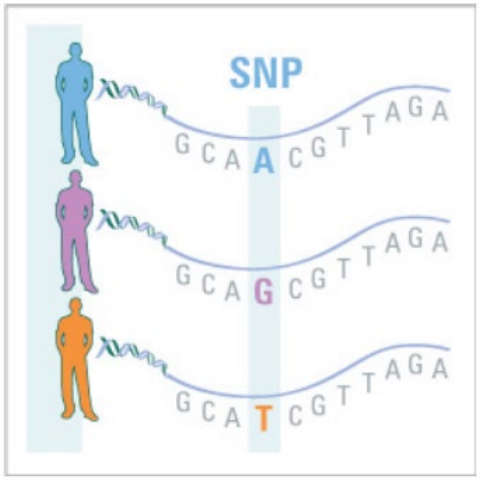


Figure 2: SNP in genetic data.

In the biological terms, particular types of variants of a SNP are named **alleles**, which describe single base variations that constitute the difference in the genomic data between two individuals of the same species.

Most SNPs are biallelic (i.e., the SNP site may be occupied by either of two different bases), and the less common base must have a frequency of at least 1%, else it is classed as a rare mutation rather than a SNP.

A single ordered set of DNA variations (SNPs) on a single chromosome is denominated as **haplotype**, and they are likely to be inherited together from a single parent, whereas the **genotype** describes the unordered combinations of alleles on both chromosomes, forming the genomic data that is employed in personalized medicine, along with many other fields.

Genomic data are primarily used in big data processing and analysis techniques. Such data are gathered by a bioinformatics system or a genomic data processing software. Typically, genomic data are processed through various data analysis and management techniques to find and analyze genome structures and other genomic parameters. Data sequencing analysis techniques and variation analysis are common processes performed on genomic data [24].

2.1.2 High-throughput methods

The aim of genomic data analysis is to determine the functions of specific genes.

In order to be able to study and investigate genomic data, firstly a compilation of the information of the genome must be done.

For this purpose, different methods have been elaborated which aim to quantify or locate all or most of the genome that harbors the biological feature (expressed genes, binding sites, etc.) of interest. These methods are known as **High-throughput methods**. Most of the methods rely on some sort of enrichment of the targeted biological feature. For example, if you want to measure expression of protein coding genes you need to be able to extract mRNA molecules with special post-transcriptional alterations that protein-coding genes acquire, as done in many RNA sequencing (RNA-seq) experiments. This part depends on available molecular biology and chemistry techniques, and the final product of this part is RNA or DNA fragments.

Then you need to be able to tell where these fragments are coming from in the genome and how many of them there are. One method employed for this consists of microarrays, in which one had to design complementary bases, called “oligos” or “probes”, to the genetic material enriched via the experimental protocol. If the enriched material is complementary to the oligos, a light signal will be produced, and the intensity of the signal will be proportional to the amount of the genetic material pairing with that oligo. There will be more probes available for hybridization (process of complementary bases forming bonds), so the more fragments available, stronger the signal. For this to be able to work, you need to know at least part of your genome sequence, and design probes. If you want to measure gene expression, your probes should overlap with genes and should be unique enough to not to bind sequences from other genes.

This technology is now being replaced with sequencing technology known as **High-throughput sequencing**, where you directly sequence your genetic material. If you have

the sequence of your fragments, you can align them back to the genome, see where they are coming from, and count them. This is a better technology where the quantification is based on the real identity of fragments rather than based on hybridization to designed probes [12].

In summary, HT techniques have the following steps, summarized in Figure 3:

- Extraction: This is the step where you extract the genetic material of interest, RNA or DNA.
- Enrichment: In this step, you enrich for the event you are interested in. For example, protein binding sites. In some cases, such as whole-genome DNA sequencing, there is no need for enrichment step. You just get fragments of genomic DNA and sequence them.
- Quantification: This is where you quantify your enriched material. Depending on the protocol you may need to quantify a control set as well, where you should see no enrichment or only background enrichment.

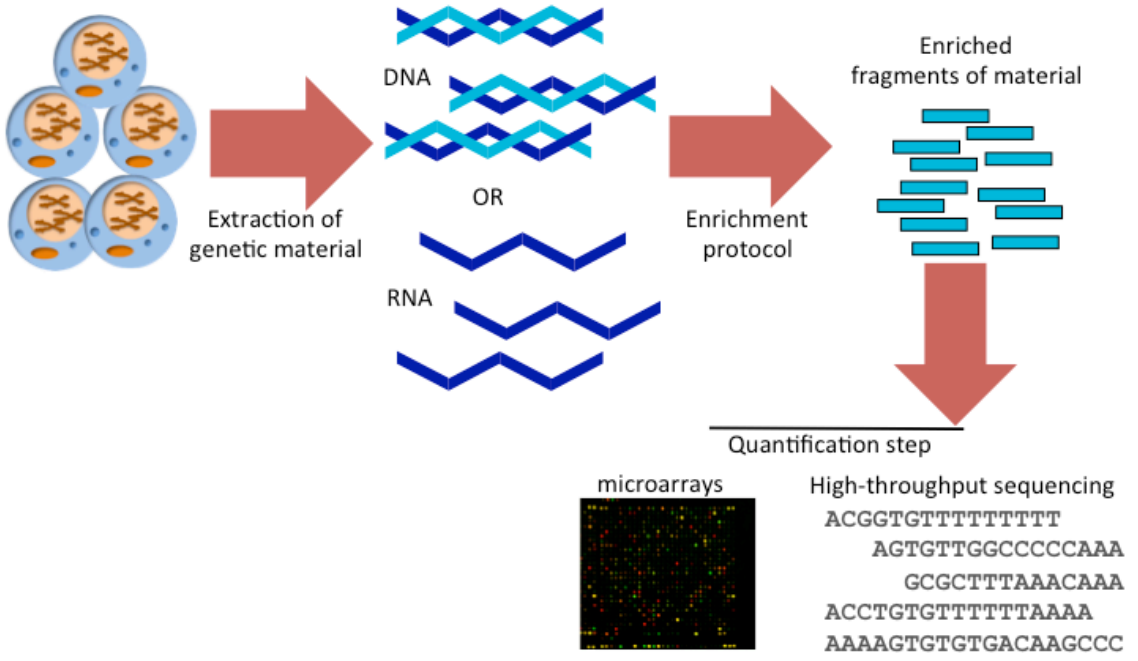


Figure 3: Steps of high-throughput assays in genome biology.

High-throughput sequencing, or massively parallel sequencing, is a collection of methods and technologies that can sequence DNA thousands/millions of fragments at a time, in contrast to older technologies that can produce a limited number of fragments at a time.

As of data repositories for genomics, there are two requirements to be able to visualize genomes and their associated data:

1. We need to be able to work with a species that has a sequenced genome

2. We want to have annotation on that genome, meaning, at the very least, you want to know where the genes are.

Most genomes after sequencing are quickly annotated with gene-predictions or known gene sequences are mapped on to them, and you can also have conservation to other species to filter functional elements. If you are working with a model organism or human, you will also have a lot of auxiliary information to help demarcate the functional regions such as regulatory regions, ncRNAs, and SNPs that are common in the population. Or you might have disease- or tissue-specific data available. The more the organism is worked on, the more auxiliary data you will have.

As for this study the security of human genomic data is the main objective, a broad set of auxiliary data is also compiled for the human genomics.

The data analysis steps typically include data collection, quality check and cleaning, processing, modeling, visualization, and reporting.

1) Data collection: In genomics, data collection is done by high-throughput assays.
2) Data quality check and cleaning: It is common to have missing values or measurements that are noisy. Data quality check and cleaning aims to identify any data quality issue and clean it from the dataset.
3) Data processing: Refers to processing the data into a format that is suitable for exploratory analysis and modeling. Oftentimes, the data will not come in a ready-to-analyze format. You may need to convert it to other formats by transforming data points (such as log transforming, normalizing, etc.), or subset the data set with some arbitrary or pre-defined condition [4]. In terms of genomics, processing includes multiple steps.
4) Exploratory data analysis and modeling: Takes in the processed or semi-processed data and applies machine learning or statistical methods to explore the data. For example, the exploration's aim can be to observe if the samples are grouped as expected by the experimental design, or if there are outliers or any anomalies. Then, a modeling step is done. This generally refers to modeling your variable of interest based on other variables you measured. In the context of genomics, it could be that you are trying to predict disease status of the patients from expression of genes you measured from their tissue samples. Then your variable of interest is the disease status. This kind of approach is generally called "predictive modeling" and could be solved with regression-based machine learning methods, like linear regression or other statistical methods.

This final data is what composes the data in genomic databases, such as the National Library of Medicine (NIH), which contains genome information, such as sequences, maps, chromosomes, assemblies, and annotations for numerous species, like humans, dogs, or numerous bacteria.

Some of these databases have compiled genetic, genomic, and multi-omic information about humans to develop databases that can aid the research of diseases, helping investigators understand, diagnose, and treat diseases according to the person's genomic data, constituting the **personalized medicine** model.

Personalized medicine, or precision medicine consists of a set of practices that uses an individual's genetic profile to guide decisions made in regard to the prevention, diagnosis and treatment of a particular disease, choosing the proper medication or therapy which can yield the best results for a particular individual.

However, these databases are composed of highly sensitive data that is tightly linked to the correct exploitation of molecular data associated with a genome sequence, which in consequence, can leak information about a person, being possible to reidentify an individual from a specific genomic data set, which reveals the current association or future susceptibility of some diseases for that individual, which results in a privacy violation, so means to anonymize and efficiently apply privacy-preserving techniques, while avoiding hampering research are needed.

2.1.3 Storing genomic sequence data

The raw genomic sequence of a single individual can be encoded in 1.5GB by representing each of the four symbols (nucleotides) of the genetic alphabet as two-bit values. Therefore, taking into account that one genome copy of a human being (23 chromosomes) is roughly 3 billion symbols long and that each individual inherits two such copies from their two parents, the amount of space that genomic data requires to be stored is considerably high.

Moreover, the current state of technology does not allow error-free determination of the two genomic copies in full-length chromosomes from a single molecule. Instead, millions of DNA molecules are obtained (e.g., from a blood sample), broken into small fragments of a few hundred nucleotides and the resulting "library" of fragments is sequenced yielding billions of so-called "reads" of between 100 and 1000 nucleotides. These are stored together with technical quality information (i.e., the estimated reliability of the value) using the FASTQ format. The size of a FASTQ file is in the order of tens of gigabytes.

The FASTQ format are the output obtained from sequencing facilities in text file, originated from the FASTA format [22].

This format is designed to handle base quality metrics output from sequencing machines. In this format, both the sequence and quality scores are represented as single ASCII characters. The format uses four lines for each sequence, and these four lines are stacked on top of each other in text files output by sequencing workflows. Each of the 4 lines will represent a read.

- The first line begins with the '@' character and is followed by a sequence identifier and an optional description. This line is utilized by the sequencing technology, and usually contains specific information for the technology. It can contain flow cell IDs, lane numbers, and information on read pairs.
- The second line is the sequence letters that identify the nucleotides of the chain.
- The third line begins with a '+' character; it marks the end of the sequence and is optionally followed by the same sequence identifier again in the first line.

- The fourth line encodes the quality values for the sequence in the second line and must contain the same number of symbols as letters in the sequence. Each letter corresponds to a quality score. Although there might be different definitions of the quality scores, a de facto standard in the field is to use "Phred quality scores". These scores represent the likelihood of the base being called wrong. These quality scores are computed as follows:

$$Q_{pbred} = -10 * log_{10}e$$

where e is the probability that the base is called wrong. Since the score is in minus log scale, the higher the score, the more unlikely that the base is called wrong.

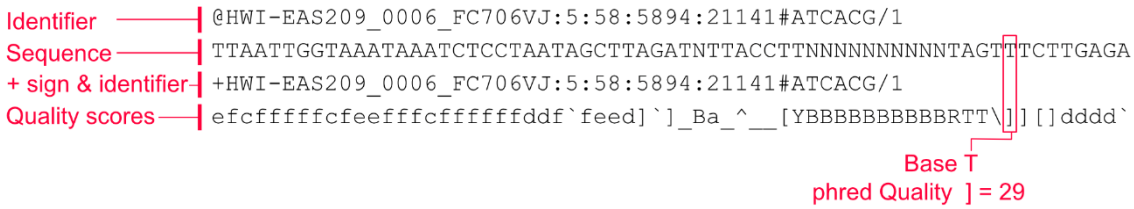


Figure 4: FASTQ file format.

ASCII_BASE=33 Illumina, Ion Torrent, PacBio and Sanger

Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII
0	1.00000	33 !	11	0.07943	44 ,	22	0.00631	55 7	33	0.00050	66 B
1	0.79433	34 "	12	0.06310	45 -	23	0.00501	56 8	34	0.00040	67 C
2	0.63096	35 #	13	0.05012	46 .	24	0.00398	57 9	35	0.00032	68 D
3	0.50119	36 $	14	0.03981	47 /	25	0.00316	58 :	36	0.00025	69 E
4	0.39811	37 %	15	0.03162	48 0	26	0.00251	59 ;	37	0.00020	70 F
5	0.31623	38 &	16	0.02512	49 1	27	0.00200	60 <	38	0.00016	71 G
6	0.25119	39 '	17	0.01995	50 2	28	0.00158	61 =	39	0.00013	72 H
7	0.19953	40 (	18	0.01585	51 3	29	0.00126	62 >	40	0.00010	73 I
8	0.15849	41)	19	0.01259	52 4	30	0.00100	63 ?	41	0.00008	74 J
9	0.12589	42 *	20	0.01000	53 5	31	0.00079	64 @	42	0.00006	75 K
10	0.10000	43 +	21	0.00794	54 6	32	0.00063	65 A			

ASCII_BASE=64 Old Illumina

Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII
0	1.00000	64 @	11	0.07943	75 K	22	0.00631	86 V	33	0.00050	97 a
1	0.79433	65 A	12	0.06310	76 L	23	0.00501	87 W	34	0.00040	98 b
2	0.63096	66 B	13	0.05012	77 M	24	0.00398	88 X	35	0.00032	99 c
3	0.50119	67 C	14	0.03981	78 N	25	0.00316	89 Y	36	0.00025	100 d
4	0.39811	68 D	15	0.03162	79 O	26	0.00251	90 Z	37	0.00020	101 e
5	0.31623	69 E	16	0.02512	80 P	27	0.00200	91 [	38	0.00016	102 f
6	0.25119	70 F	17	0.01995	81 Q	28	0.00158	92 \	39	0.00013	103 g
7	0.19953	71 G	18	0.01585	82 R	29	0.00126	93]	40	0.00010	104 h
8	0.15849	72 H	19	0.01259	83 S	30	0.00100	94 ^	41	0.00008	105 i
9	0.12589	73 I	20	0.01000	84 T	31	0.00079	95 _	42	0.00006	106 j
10	0.10000	74 J	21	0.00794	85 U	32	0.00063	96 `			

Figure 5: Commonly used Phred quality scores in different sequencing techniques.

To solve the presented size problem that appears in extended sequences represented in a file, given a known reference sequence, it is sufficient to only encode the difference between an individual's sequence and the reference, storing only the variants found in the individual in VCF files that amount to tens of megabytes in size.

However, these files, while reducing the storage size considerably, leak important information, as the size can be correlated to the number of variants found in the individual.

Person-specific genomic records must be shared in a manner that preserves the anonymity of the data subjects. This requirement is rooted in both social concerns and public policy. Many people fear that sensitive information learned from their medical and genomic records will be misused or abused. To mitigate such concerns, many countries have enacted policies that limit the sharing of a subject's genomic information in a personally identifiable form. For this, several institutes, such as the National Institutes of Health (NIH) are drafting policy that will require scientists to share genomic data studied with NIH funding once "identifiable information" has been removed.

2.1.4 Privacy in genomic data

In order to comply with newly stablished policies, investigators, at the completion of their studies, need to share their data collections to a centralized repository, so that other researchers can perform scientific investigations on the integrated data.

To summarize the problem, genomic data collectors need to satisfy two goals when sharing genomic data:

1. Data utility: the data should be useful for scientific investigations.
2. Data privacy: the data should not reveal the subjects' identities.

These two problems can be solved following the presentation on ideas for dissemination of genetic data by Anna Oganian, who discusses novel techniques to solve the privacy problems in the dissemination of genetic data by employing synthetic data.

Instead of working with original genetic data of subjects, that would need otherwise to be modified or tampered with to maintain privacy, we can achieve k-anonymity by employing synthetic microdata generation techniques applied to the genetical field to generate synthetic data, which constitutes as a low-risk alternative to perturbing/masking the original data, as synthetic records hold no direct correspondence to the original records.

For this, we can construct new genetic data sets following the principles of population genetics by mixing and hashing initial genetic material over many generations in order to create a genetically disperse future population that shares the same genetic traits in general as the initial population, but holds no direct correspondence, by simulating the recombinations and crossovers the genetic material is subjected to when crossing individual:

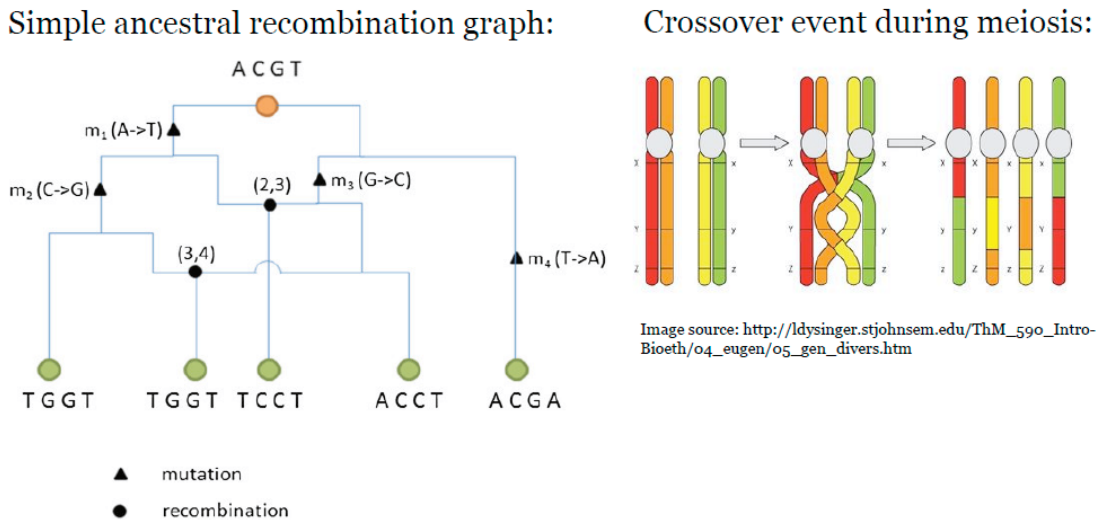


Figure 6: Recombination and crossover schemes of genetic material.

In order to synthesize new genotypic data, we first need to understand how it is conformed. Genotype data is usually presented in a form of genotypes describing SNPs on both chromosomes inherited from both parents, which are unordered alleles on both chromosomes comprising individual genotypes. Then, haplotypes are two ordered sequences of alleles that have been inherited together from the parents.

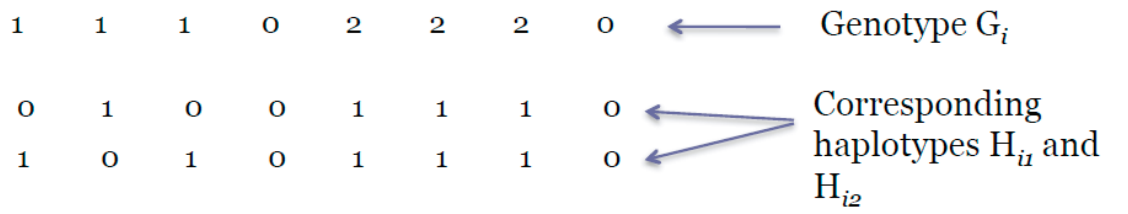


Figure 7: Genotype and haplotype example.

Assuming that we have L biallelic SNPs and that the two alleles at each SNP are coded 0 and 1, let H be a set of haplotypes at these L SNPs and G the set of genotype data at the L SNPs in N individual, with $G_i = \{G_{i1}, G_{i2}, \ldots, G_{iL}\}$ denoting the genotypes of the i^{th} individual. Then, to synthesize new genotypes based on the original genotypes, we can employ the following algorithm:

1. For each individual genotype G_i statistically estimate a pair of corresponding haplotypes H_{i1} and H_{i2} using haplotype estimation methods. Haplotype estimation methods are based in the Hidden Markov Model (HMM) which models the haplotypes underlying G_i as an imperfect mosaic of haplotypes from a reference set.
   There are software that already estimate the haplotypes for the whole chromosomes, such as SHAPEIT, a free software that holds a linear complexity with the number of SNPs.
2. After generating the haplotypes, we randomly pair them to form new synthetic genotypes, pairing haplotypes from two different individuals, to simulate their crossover. These synthetic genotypes represent new non-existing synthetic individual genomic data. We can conserve the statistical data of the alleles if each

estimated haplotype contributes exactly to one synthetic individual.

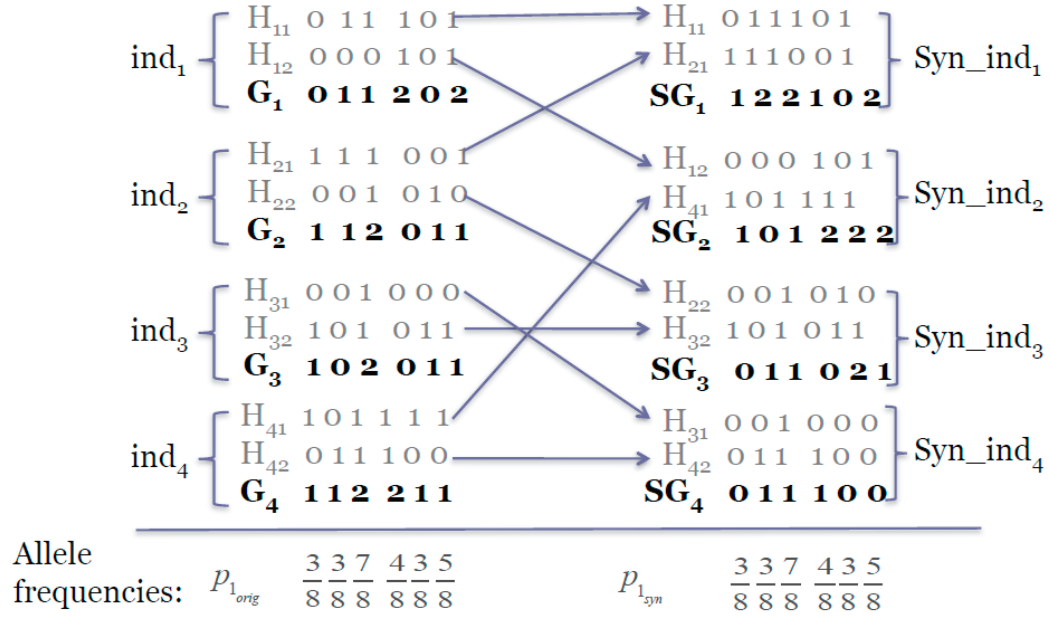


Figure 8: Allele frequencies preservation in synthetic genomic microdata generation.

This way, we can preserve the allele frequencies while avoiding re-identification.

3. We can simulate crossover and mutation events by crossing haplotypes, which will not change allele frequency (but will enhance diversity in genotypes in future generations), and modifying one allele to add mutations, which will slightly change the frequencies.

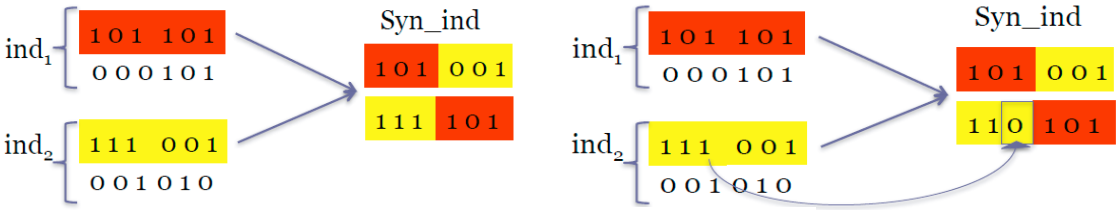


Figure 9: Crossover and mutation events simulation.

Differences in allele frequencies between the synthetic data and the original data may be desirable, as it reduces the disclosure risk of the information.

2.1.5 Privacy problems to be solved

Privacy-preserving sharing of genomic data: Securing against reidentification attacks, as genomic data withstands as extremely sensitive data, so not allowing public sharing of these data without any form of privacy guarantee or anonymization is mandatory [37]. Ex: Homomorphic encryption, Differential privacy.

Secure computation and storage of genomic data: It connotes the leakage risks from the storage or computation of genomic data. Allowing third parties to compute on the data in plaintext involves unwanted risks and possible leaks from the storage. This problem can be affronted by employing Homomorphic encryption, Garbled circuit or Secure hardware.

Query or output privacy: The output of any query from the genomic data, even after applying privacy-preserving techniques, can reveal the researcher's interest and some data characteristics. These problems are acquiring increasing importance lately in the past years, due to cases such as the attacks against the aggregated results of Genomic Beacon Service, which further elevate the necessity of this privacy. Therefore, applying privacy to the queries and their outputs is mandatory, as they are as sensitive as they data, revealing, for example, the presence of an individual or a certain group in a genomic study. For this problem, techniques such as Differential privacy can be applied.

2.2 Privacy preserving techniques

2.2.1 K-anonymity

The concept of k-anonymity was introduced into information security and privacy back in 1998 built on the idea that by combining sets of data with similar attributes, identifying information about any one of the individuals contributing to that data can be obscured. It aims to hide an individual's data in a pool of a larger group data, meaning the information in the group could correspond to any single member, thus masking the identity of the individual or individuals in question.

The k in k-anonymity refers to a variable, the number of times each combination of values appears in a data set. If k=2, the data is said to be 2-anonymous. This means the data points have been generalized enough that there are at least two sets of every combination of data in the data set. This means the data points have been generalized enough that there are at least two sets of every combination of data in the data set.

k-Anonymity main's purpose is to protect against hackers or malicious parties using 're-identification,' tracing data's origins back to the individual it is connected to in the real world, thus learning sensitive data for an individual, compromising their privacy.

It can be implemented through generalization, suppression, minimizing risk among other techniques.

2.2.2 Homomorphic encryption:

Homomorphic encryption (HE) [26] algorithms are algorithms that provide forms of encryptions with an additional evaluation capability for computing over encrypted data without access to the secret key, allowing mathematical operations to be performed on encrypted data by mapping a set of operation on cleartext to another set of operations on ciphertext. HE schemes can be classified into different forms, such as Fully, Leveled and SomeWhat Homomorphic Encryption schemes, which mainly differ in compactness, correctness, or the functions they can compute.

- Fully Homomorphic Encryption (FHE): Allows operating any arithmetic circuit to be applied to ciphertexts in order to obtain the encryption of the result that would be obtained if applying the same circuit to the cleartext.
- Leveled Fully Homomorphic Encryption (Leveled-FHE): It restricts certain depths, computed by the multiplication gates, to remove (or spread) the bootstrapping operation in order to gain performance, as well as removing the

assumption of circular security (Which states that one can safely encrypt its private key with their own public key).
- SomeWhat Homomorphic Encryption (SHE) [27]: Allow to evaluate only circuits of multiplicative depth, thus reducing the number of restrictions and increasing the efficiency compared to FHE.

Applied schemes:

- Additive HE (Paillier encryption) on a semi-honest cloud server (*Kantarcioglu et al.*) to securely store the genomic sequences in a cloud server, encoding the genomic sequences with a binary representation and then encrypted the individual encoding by Paillier encryption, which consists of an additive homomorphic.
- Additive HE, privacy-preserving genetic risk calculation (*Ayday et al.*), which depended on the DGK (Damgãrd, Geisler and Krøigaard) cryptosystem to safely realize computations on integers.
- Applying secure statistical algorithms, such as Hardy-Weinberg equilibrium and linkage disequilibrium, while using fully homomorphic encryption to allow any possible computation on the encrypted data.
- The most used scheme is the SomeWhat HE in genomic data, covering homomorphically computing logistic regression [27].

HE is not efficient enough for large-scale generic genomic data computation for the first two problems, as multiplication or bootstrapping operations take too much time to use it realistically on complex functions (like machine learning) [33].

The CKKS scheme is an FHE scheme supporting the arithmetic operations on encrypted data over real or complex numbers. Any users with the public key can process the encrypted real or complex data with the CKKS scheme without knowing any confidential information. The security of the CKKS scheme is based on the Ring-LWE hardness assumption. The supported homomorphic operations are the addition, the multiplication, the rotation, and the complex conjugation operation, and each operation except the homomorphic rotation operation is applied component-wisely.

2.2.3 CKKS

The Cheon-Kim-Kim-Song (CKKS) scheme was firstly introduced in the paper "*Homomorphic Encryption for Arithmetic of Approximate Numbers*".

It is an FHE scheme that supports arithmetic operations to be performed on encrypted data over vectors of real or complex numbers, where any users with the public key can process the encrypted real or complex data with the CKKS scheme without knowing any confidential information.

The supported homomorphic operations are the addition, the multiplication, the rotation, and the complex conjugation operation, and each operation except the homomorphic rotation operation is applied component-wisely. It is one of the enhancements this scheme presents over the original FHE scheme.

This scheme executes the rotation operation homomorphically by performing a cyclic shift of the vector by some step, while the multiplication, rotation, and complex conjugation operations in the CKKS scheme need additional corresponding evaluation keys and the key-switching procedures.

The CKKS scheme employs polynomials in the operations because they provide a good trade-off between security and efficiency as compared to standard computations on vectors. The central idea to implement a homomorphic encryption scheme is to have homomorphic properties on the encoder, decoder, encryptor and decryptor. This way, operations on ciphertext will be decrypted and decoded correctly and provide outputs as if operations were done directly on plaintext.

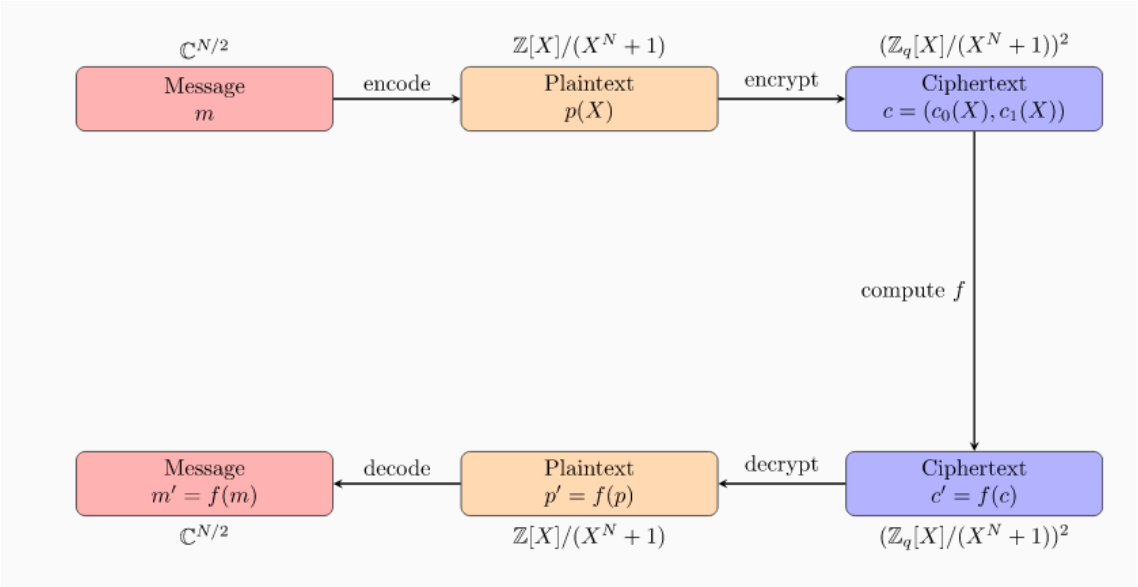


Figure 10: High level view of CKKS.

The figure 10 represents a high-level view of the CKKS scheme, where m represents the message, a vector of values on which we want to perform certain computations. It is encoded into a plaintext polynomial p(X), and then encrypted with the public key to create c, composed by a couple of polynomials.

The operations that the scheme can employ are applied to this ciphertext, allowing addition, multiplication, and rotation. These operations are denoted by the function f.

Finally, decrypting c' = f(c) with the secret key will yield p' = f(p), and once decoded, we will get m = f(m), the resulting message of the operation applied.

2.2.4 RNS-CKKS

The RNS-CKKS scheme is a modern implementation of the CKKS scheme which solves the main problem of the CKKS, since the original CKKS scheme needs big integers, so it employs a multi-precision library, which requires a higher computational complexity.

The RNS-CKKS scheme solves this problem by employing the residue number of the integers, where the big integer is split into several small integers, and the addition and the multiplication of the original big integers are equivalent to the corresponding component-wise operations of the small integers.

Each real number data is scaled with some big integer, called the scaling factor, and then rounded to the integer before encrypting the data. When the two data encrypted with the CKKS scheme are multiplied homomorphically, the scaling factors of the two data are also multiplied. This scaling factor should be reduced to the original value using the rescaling operation by employing the representational abilities of the Residual Neural Network (ResNet), more exactly, the ResNet-20 [2] (using 20 parameter layers), which employs the following specifications:

Layer		Input Size	#Inputs	Filter Size	#Filters	Output Size	#Outputs
Conv1		32 × 32	3	3 × 3	16	32 × 32	16
Conv2	2-1	32 × 32	16	3 × 3	16	32 × 32	16
	2-2	32 × 32	16	3 × 3	16	32 × 32	16
	2-3	32 × 32	16	3 × 3	16	32 × 32	16
Conv3	3-1-1	32 × 32	16	3 × 3	32	16 × 16	32
	3-1-2	16 × 16	32	3 × 3	32	16 × 16	32
	3-1-s	32 × 32	16	1 × 1	32	16 × 16	32
	3-2	16 × 16	32	3 × 3	32	16 × 16	32
	3-3	16 × 16	32	3 × 3	32	16 × 16	32
Conv4	4-1-1	16 × 16	32	3 × 3	64	8 × 8	64
	4-1-2	8 × 8	64	3 × 3	64	8 × 8	64
	4-1-s	16 × 16	32	1 × 1	64	8 × 8	64
	4-2	8 × 8	64	3 × 3	64	8 × 8	64
	4-3	8 × 8	64	3 × 3	64	8 × 8	64
Average Pooling		8 × 8	64	8 × 8	64	-	64
Fully Connected		64 × 1	1	-	-	-	10

Figure 11: The specification of ResNet-20 (CIFAR-10)

By employing the RNS-CKKS scheme to a standard deep neural network the efficiency of the neural network can be incremented greatly, removing the need of using big integers, increasing the computational complexity, while achieving the same accuracy than the original neural networks, adapting deep learning techniques to grand quantities of data.

2.2.5 ResNet

A Residual Neural Network (ResNet) [2] is an artificial neural network that conforms a gateless (open-gated) variant of the HighwayNet (The first working deep feedforward neural network, composed by hundreds of layers by employing skip connections modulated by learned gating mechanisms that regulated information flow).

The ResNet is characterized due to applying identity mapping, where the input to some layer is passed directly or as a shortcut to some other layer by gates. These open gates employ skip connections (or shortcuts) to jump over layers by employing an additional weight matrix for the gates, and typically are implemented with double or triple layer skips with nonlinearities (ReLU) and batch normalization in between.

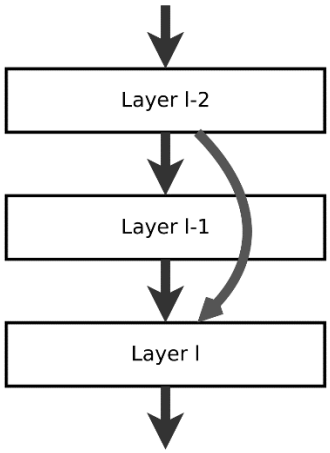


Figure 12: Canonical form of a residual neural network. A layer $\ell - 1$ is skipped over activation from $\ell - 2$.

Skip connection is basically the identity mapping where the input from previous layer is added directly to the output of the other layer. Skipping connections is desirable, as it can be used to avoid the problem of vanishing gradients, allowing the optimization of neural networks to be less time-consuming (as the gating mechanisms facilitate the information flow across many layers) and mitigates the accuracy saturation problem (Degradation), which arises when employing a numerable amount of layers (adding more layers to a suitably deep model leads to higher training error).

During training, the weights adapt to mute the upstream layer, and amplify the previously skipped layer, thus training the skip connections in the same process.

Thanks to the employment of the ResNet architecture, the performance of neural networks with a large number of layers have increased substantially, allowing the use of neural networks for more complex situations, such as genomic research.

2.2.6 Pseudorandom function

Pseudorandom functions (PRFs) [32] were first introduced by Goldreich et al. PRFs are deterministic functions of a key and an input that is computationally indistinguishable from a truly random function of the input out of all the set of functions having the same domain and range. For all inputs x, the output of a PRF(k,x) is indistinguishable and random to the eyes of a computationally bounded adversary.

Let s be a security parameter, let K be a key of length s bits, and let f(K,x) be a function on keys K and inputs x. Then, f is a PRF if:

- f can be computed in polynomial time in s.
- If K is random, then f cannot be distinguished from a random function in polynomial time.

Distinguishability refers to the ability of an algorithm to tell whether a function is not truly random. Let g be a truly random function of x with the same output length as f. Supposing a polynomial-time algorithm A is given access to an oracle that, on input x, either consistently returns f(K,x) or consistently returns g(x).

After some polynomial number of accesses to the oracle, the algorithm outputs a guess, b, as to whether the oracle is f or g.

Let ϵ be A's difference in probabilities (advantage)

$$\epsilon = |Pr[b = "f" \mid oracle\ is\ f] - \Pr[b = "f" \mid oracle\ is\ g]$$

If the inverse 1/ ϵ grows faster than any polynomial in s for all polynomial-time algorithms A, then the function f is said to be indistinguishable from a random function.

Pseudorandomness is important in privacy studies as it is a stronger requirement than being a one-way function (functions that are easy to compute on every input but hard to invert given the image of a random input), since a one-way function only needs to be hard to invert, whereas a PRF also needs to be hard to guess when the key is used, thus providing a way to turn an input into a value that is effectively random.

This is helpful for computing MAC algorithms, deriving keys from other keys, and more generally for replacing random number generators in an application with a deterministic function, when a secret key is available.

2.2.7 Message Authentication Codes

Messages Authentication Codes (MACs) [30], or tags, are short pieces of information additionally added to messages employed to authenticate the origin and nature of a message. MACs employ authentication cryptography to verify the legitimacy of data sent through a network or transferred from one person to another.

A strong MAC can help preventing an adversary from modifying a message sent by one party to another, without the parties detecting that a modification has been made.

The first step in the MAC process is the establishment of a secure channel between the receiver and the sender. To encrypt a message, the MAC system uses an algorithm, which uses a symmetric key and the plain text message being sent. The MAC algorithm then generates authentication tags of a fixed length by processing the message.

For this, a sender associates a message m with a MAC tag t that was generated by computing MAC(k,m) with a key k. The resulting computation is the message's MAC.

This MAC is then appended to the message and transmitted to the receiver, who also knows the key, can verify whether the tag is correct on the associated message by checking if MACVfy(k,m,t) = 1. If the resulting MAC the receiver arrives at equals the one sent by the sender, the message is verified as authentic, legitimate, and not tampered with.

In effect, MAC uses a secure key only known to the sender and the recipient. Without this information, the recipient will not be able to open, use, read, or even receive the data being sent. If the data is to be altered between the time the sender initiates the transfer and when the recipient receives it, the MAC information will also be affected.

Therefore, when the recipient attempts to verify the authenticity of the data, the key will not work, and the end result will not match that of the sender. When this kind of discrepancy is detected, the data packet can be discarded, protecting the recipient's system.

A one-time MAC denoted by OTMAC is a restricted version of a MAC, where each key can used only once to compute a MAC tag, providing stronger security than a normal MAC.

The strong security of a OTMAC scheme implies that no adversary can forge a new valid message-tag pair and employing PRFs to derive keys for new OTMAC functions can diminish the computational times, erasing the need to generate new keys.

2.2.8 Oblivious Transfer

Oblivious Transfer (OT) [31], firstly introduced by Michael O. Rabin, is a cryptographic protocol in which a sender transfers one of potentially many pieces of information to a receiver but remains oblivious as to what piece has been transferred, if any has been sent.

In this protocol, the sender has two input messages m0 and m1, and the receiver has an input bit b. At the end of the protocol the receiver learns mb and nothing else, while the sender does not know which message the receiver has obtained, although the protocol can be employed to select out of many other messages (1-out-of-n oblivious transfer protocol).

The protocol starts by the sender generating n RSA key pair (pk, sk), where n is the number of messages that will be sent to the receiver and sends the public keys. The receiver selects b ϵ {0, n-1} that corresponds to the message he wants to learn about, and chooses a random symmetric key k, encrypts it with the pk, which is sent back to the sender. The sender then computes n keys with k and each message sk and creates a

ciphertext of each message $E_{kn}(mk)$, which are sent to the receiver, who will be able to only decrypt the ciphertext that corresponds to the message it desired.

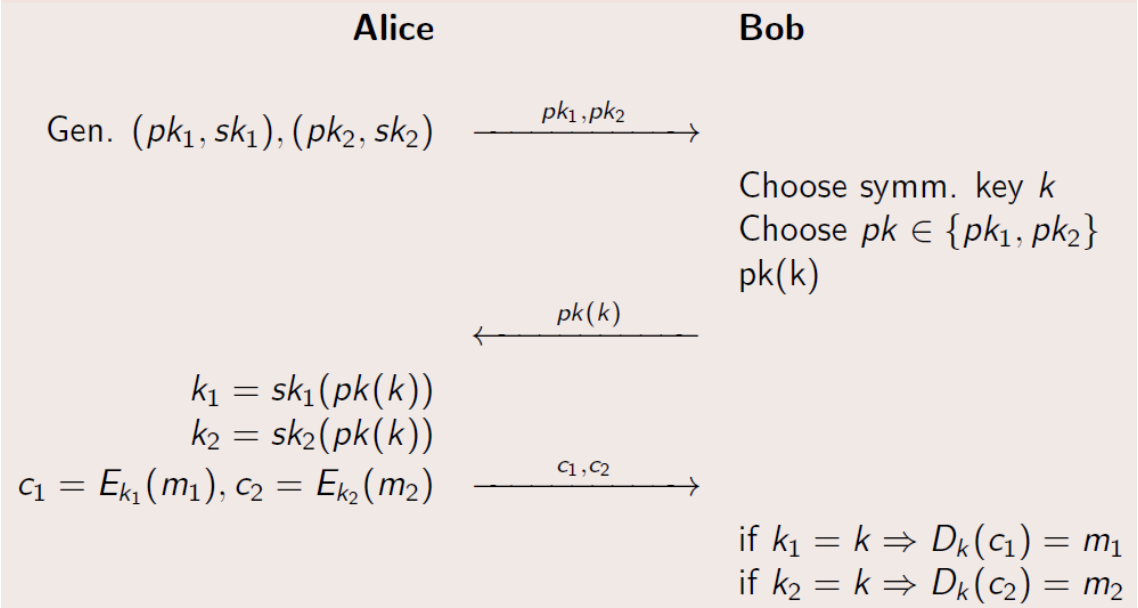


Figure 13: 1-out-of-2 OT protocol (Schneier, 1996)

The security requirement demands that the sender should not learn the bit b and the receiver should not learn anything about m_{1-b}.

2.2.9 Garbled circuit

A garbled circuit (GC) [29] is a constant round secure protocol that allows any function to be computed between multiple parties while hiding the inputs from each other, without needing the presence of a trusted third party, as was originally schemed by Andrew Yao [28].

In GCs, the function has to be described as a Boolean circuit (a mathematical model for combinational digital logic circuits, defined in terms of the logic gates they contain). This protocol is executed between two different parties, the garbler, which encrypts the circuit, and the evaluator, who decrypts the circuit and obtains the encrypted outputs.

The protocol consists of 5 steps:

1. Circuit generation: The underlying function that wants to be executed is formed by generating a Boolean circuit, which will be employed in the protocol.
2. Garbling: The garbler encrypts the Boolean circuit (also known as garbling the circuit). For it, the garbler assigns two randomly generated strings, called labels, to each wire in the circuit: one for Boolean semantic 0 and one for 1. (The label is k-bit long where k the security parameter and is usually set to 128.) and then replaces 0 and 1 in the truth tables with the corresponding labels for each gate in the circuit, forming a truth table. This truth table is encrypted with a double-key symmetric encryption function, $Enc_k(X)$, where k is the secret key and X are the values to be encrypted.
   The last step of the garbling process consists of a permutation of the table, in order to make the output values undeterminable by the row.
3. Data transfer: The garbler sends the computed garbled tables for all gates in the circuit to the evaluator, as well as the tables that correspond to the garblers input. The evaluator needs the input tables to open the garbled tables, thus needing the garbler's tables of their inputs. As the labels are randomly generated by the garbler, the evaluator will not learn anything about the garbler's input. (For example, if the garbler's input is $a_4a_3a_2a_1$, where a is their input, she sends the corresponding labels for each input, thus not leaking information).
   The labels that correspond to the evaluator are sent through oblivious transfer for each bit of his input, which disables the garbler to learn anything of the evaluator's inputs.
4. Evaluation: The evaluator, who now has in possession the garbled tables and the input labels, can decrypt the rows of the garbled tables through all the gates, retrieving for one row for each table the corresponding output label (a is the garbler's input and b is the evaluator's input), until it reaches the output labels:
$$X^c = Dec_{x^a,x^b}\left(garbled_{table[i]}\right)$$
5. Revealing output: After the evaluation process has been completed, the evaluator obtains the output label, and the garbler knows its mapping to Boolean value, as she has both labels that were computed. At this point, either the garbler can share the label information to the evaluator, or the evaluator can reveal the output to the garbler so that one or both of the parties can learn the output.

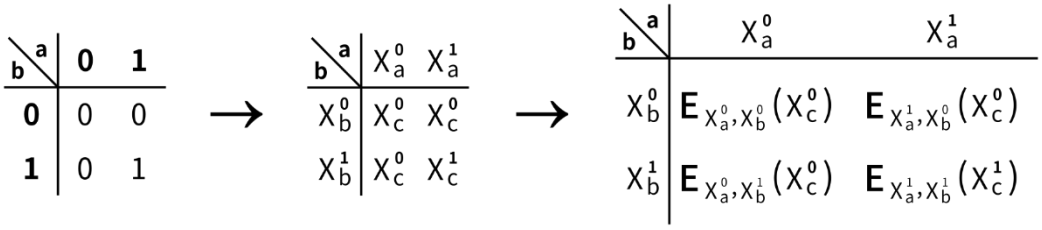


Figure 14: Example of the construction of the garbled table

One of the major advantages of garbled circuits is that two different parties can evaluate a function without revealing any information about their input aside from trivial info leaked by the output of the garbling scheme, due to its properties, as it is based on the oblivious transfer protocol, where a string is transferred between a sender and a receiver out of many possible different pieces of information so that the receiver doesn't know what the sender is sending, remaining oblivious of the real information. In this, a sender has two strings S_0 and S_1. The receiver chooses i in {0, 1} and the sender sends S_i with the oblivious transfer protocol such that:

1. The receiver does not gain any information about the unsent string $S_{(1-i)}$.
2. The value of i is not exposed to the sender.

Garbled circuits can be used to solve the problems that surge with the query and output privacy, and applying secure computations and storage to genomic data, as with the latest improvements they have gone through, they are optimized enough to be efficient for practical applications. However, GC, as the current state of the art, still need to be tested on large-scale genomic data computation, as realistic scenarios dictates long genomic sequences, whereas all the recent secure edit distance approximations only take small sequence lengths into account, as well as possibly requiring greater network overheads for large-scale circuit computation.

A new surging technology that can be employed for secure multiparty computation protocols are the Secret Sharing Schemes, which can be employed to solve these limitations on GCs in the genomic data ambit.

2.2.10 METIS system

The major challenge in secure computation on genomic data is performance, since human genomic data is large (as stated before).

In order to solve the second and third proposed problems, a system that gives the data-owners in full control over the disclosed information, while not imposing high computational burdens by cryptographic schemes is the most ideal solution.

For this purpose, we can elaborate a system based on the METIS system by *Dominic, Deuber et al. (2018) [41]*, which gives the data owners complete control over the genomic data, while reducing the computational costs, this system divides the structure into four distinct roles so that the computing time of data owners and clients can be reduced, as well as storage space.

The METIS system consists of a service provider system between genetic researchers and data owners, while securing the genetic data in the METIS servers, providing computations on large sets of genetic samples in a remote, secure way. For it, it designs

four different types of parties: sequencing center, server, data owners and clients, which serves as ways to ease the computation times and storage weights on the clients and data owners, as well as keeping the data secure and accessible, thanks to the employed protocol for data encoding, which leverages the security and integrity properties of garbled circuits, while allowing secure computation of functions.

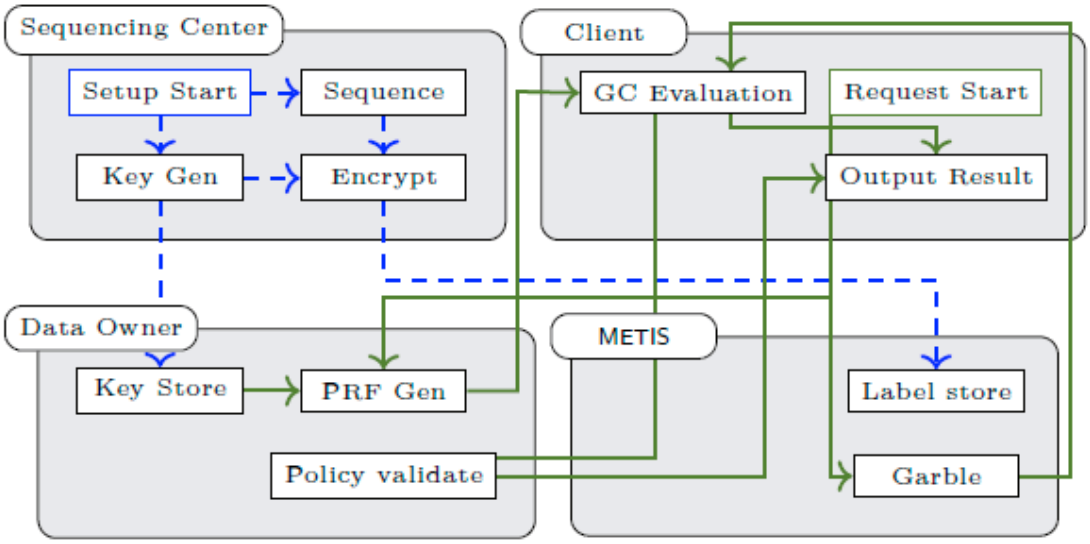


Figure 15: Structure of the METIS system

For this system, we assume the existence of a secure, authenticated channel between all parties, which are aware of the public, authenticated keys of the other parties (Using, for example PKI), or using the server as a relay and encrypting the data with the public key of the recipient, under a non-malleable encryption scheme.

The METIS protocol is structured in two phases:

- Setup phase: The sequencing center produces the DNA sequence and stores the genetic information.
- Evaluation phase: The client interacts with the server and the data owner to evaluate a function over the genetic information. The computation and communication flow must be independent of the size of the genome.

In order to keep the storage employed by the data and avoiding overburdening the owner with computation over the genome with each query, we can employ multi-party computation solutions, sharing the load of computation and communication equally among the parties. For it, a protocol for data encoding based on Garbled circuits can be employed, allowing to securely compute any function without revealing the inputs.

The **cryptographic scheme**, in the setup phase, employs a secret key, generated by the sequencing center, to evaluate a function to generate series of pseudorandom labels, which then, are used to generate a sequence of random labels of the same length, which

will be used to encode the data. The two sequences of labels constitute an encoding information, in which for every position i we store a pair of labels (l_i^0, l_i^0).

In this encoding information, if the bit of the binary representation of the sequence of the genome at position i is 0, then l_i^0 is a pseudorandom label, whereas l_i^1 is a random label, and vice versa. The sequencing center sends the secret key to the data owner. Such an encoding bears a crucial property: Given the key, one can easily re-evaluate the pseudorandom function and reconstruct the labels corresponding to the genomic sequence.

The **setup phase** consists in a series of steps in which the genomic data is produced and encoded by the system, with the collaboration of the sequencing center.

First, the sequencing center needs to have the desired data to be stored in the system, and producing a secret key, which will be employed in the cryptographic system that will encrypt the data, sampling sequences of random labels, stored in pairs, to manage the data.

Then, the sequencing center samples a random key (k) to generate the keys (k_j) which will be used for all the blocks of the sequenced DNA data (x), and creates the encoding information (e_x) as following:

For the *i*-th bit of the *j*-th block of x ($x_{j,i}$) it computes a pair of labels, which represent $x_{j,I}$ (determined by the pseudorandom function with input $k_{j,i}$)), and a randomly sampled string which represents the bit 1- $x_{j,I}$, respectively.

Consequently, the labels that correspond to the genomic data in question are fixed and can only be reconstructed using the employed pseudorandom function and the key.

Then, the sequencing center sends the encrypted data and the sequence of 0 and 1 labels for the genomic data to the server the secret key to the data owner.

This is a key aspect of the METIS system, as such an encoding bears a crucial property, where given the key, one can easily re-evaluate the pseudorandom function and reconstruct the labels corresponding to the genomic sequence. However, it does not, on its own, reveal any information about the original input (the corresponding original bitstring).

Thanks to this property, the data owner can easily decide whether a user can evaluate a function on the data easily by giving the data owner a secret key, which is computed by the sequencing center. Given the key, one can easily re-evaluate the pseudorandom function and reconstruct the labels corresponding to the genomic sequence, while not revealing any information about the original input, as the information transformed into a series of pseudorandom labels, which are sampled into sequences that can be used to reform the data.

At the end of this step, the server holds an encoding of the data, and the owner holds the corresponding decoding information, and the sequencing center can securely delete all the data, as it is stored in an encrypted manner in the server.

At the **evaluation phase** the client interacts with the server and the owner, so that the owner can grant permission to compute a function of his interest on the encrypted genomic data stored in the server.

Whenever a client wants to operate a function on a DNA sequence, the server informs the data owner about the request, while generating a garbled circuit with the encoding provided by the sequencing center and forwarding decoding information said circuit to the data owner and forwarding the circuit to the center. If the owner wants to grant access to the data for the function of the client, the owner can send the key to reconstruct the encoded input so it can evaluate the circuit, and if it wants to allow the function to be evaluated, he can also send the decoding information to the client.

This grants privacy as the output of a garbled circuit without the decoding information consists of a set of randomly chosen bitstrings, thus, maintaining the privacy of the data.

The client sends a function its query to the server, which then takes the role of the garbler, garbling the generated circuit using as input the genomic data (x) and the sent query by the client (z), returning z||f(x,z) while the client evaluates the circuit. The client does not know x thus it cannot run an oblivious transfer process for the labels that correspond to the data with the server.

Therefore, using the pseudorandom function (PRF) key that was sent by the owner, it can directly reconstruct the correct labels to evaluate the garbled circuit.

The server, after computing the result of the function requested by the client, blinds the values corresponding to the function output with a random string v, instead of sending the output table to the client (alongside with the garbled circuit), thus sending the blinded decoding information, including an OTMAC to the client and the blinding factor v together with the decoding information for the client's output to the owner.

Thanks to the OTMAC we can prevent selective failure attacks from a corrupted data owner, as any modification done the string v is noted by the client, stopping said attacks.

In parallel, the client interacts with the owner, who sends the PRF key (if so desired by the owner) that corresponds to the block of interest j, therefore, as the labels that correspond to the input data were created using this PRF key on the block key kj and the offset I, the client is now able to reconstruct the correct input labels, not leaking any additional information as the block is a random string.

Now that the client holds the output labels that correspond to z||f(x,z), he can send the request to the owner, who decodes and validates it with their policies, and, if the owner agrees to the function and the request, it can send the blinding factor to the client to verify the OTMAC and thus decoding f(x,z).

Consequently, at the end of the evaluation phase, the client only learns the output of the function f(x,z), given the case that the owner allows the execution, and the owner learns the request of the client, while not learning the result of the evaluation, covering against potential security breaches at the data owner's end.

Multiple queries from clients for the same data owner and the same data employ the same encoded data, but as the server computed different garbled circuits with the same encoded data, thus employing different encoding data for each different query.

One of the main problems that arises with the proposal of the METIS scheme consists on the use of the generic oblivious transfer technique proposed by *Gilad-Bachrach et al.* which allows to outsource computation to a semi-trusted cloud provider that has similar characteristics to other systems, but carries a linear cost in the inputs, requiring a number of oblivious transfers in size of the inputs, which makes the costs of the METIS scheme be higher the bigger the size of the inputs is. As genomic sequences are large, the METIS scheme is only suitable of peculiar cases, or for theorical use.

Sequencing Center $\mathsf{S}(x, B)$

$k \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
for $j = 1, \ldots, \left\lceil \frac{|x|}{B} \right\rceil$ **do**
  $k_j \leftarrow \mathsf{PRF}(k, j)$
  for $i = 1, \ldots, B$ **do**
    $\ell_{j,i}^{x_{j,i}} \leftarrow \mathsf{PRF}(k_j, i)$
    $\ell_{j,i}^{1-x_{j,i}} \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
$e_x \coloneqq \left\{ \left(\ell_{j,i}^0, \ell_{j,i}^1 \right) \right\}_{j \in \left[\left\lceil \frac{|x|}{B} \right\rceil \right], i \in [B]}$
return e_x, k

Data Owner $\mathsf{O}(k, \phi)$

receive $_{\mathsf{M}}(j, f, d_z, v)$
$k_j \coloneqq \mathsf{PRF}(k, j)$
send $_{\mathsf{C}}(k_j)$
receive $_{\mathsf{C}}(Y_z)$
$z \leftarrow \mathsf{De}(d_z, Y_z)$
if $z \neq \perp$ **and** $\phi(f, z) = 1$
  send $_{\mathsf{C}}(v)$
else send $_{\mathsf{C}}(\perp)$

Server $\mathsf{M}(e_x, |z|)$

receive $_{\mathsf{C}}(j, f)$
for $i = 1, \ldots, |z|$ **do**
  $\ell_i^0 \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
  $\ell_i^1 \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
  $\mathsf{sender}_{\mathsf{OT}}\left(\left(\ell_i^0, \ell_i^1 \right) \right)$
$e_z \coloneqq \left\{ \left(\ell_i^0, \ell_i^1 \right) \right\}_{i \in [|z|]}$
$e \coloneqq e_z || e_x$
$\nabla(x, z) \coloneqq z || f(x, z)$
$(F, d) \leftarrow \mathsf{Gb}\left(1^{\lambda}, \nabla(\cdot, \cdot), e \right)$
$d_z || d_{f(x,z)} \coloneqq d$
$\mathsf{sk} \leftarrow \mathsf{MACGen}(1^{\lambda})$
$t \leftarrow \mathsf{MAC}(\mathsf{sk}, d_{f(x,z)})$
$v \leftarrow\!\!\$\ \{0,1\}^{|t|+|d_{f(x,z)}|}$
$w \coloneqq v \oplus (t || d_{f(x,z)})$
send $_{\mathsf{O}}(j, f, d_z, v)$
send $_{\mathsf{C}}(F, \mathsf{sk}, w)$

Client $\mathsf{C}(j, z, f, B)$

send $_{\mathsf{M}}(j, f)$
for $i = 1, \ldots, |z|$ **do**
  $\ell_i^{z_i} \leftarrow \mathsf{receiver}_{\mathsf{OT}}(z_i)$
$\mathcal{L}_z \coloneqq \left\{ \ell_i^{z_i} \right\}_{i \in [|z|]}$
receive $_{\mathsf{O}}(k_j)$
for $i = 1, \ldots, B$ **do**
  $\ell_{j,i}^{x_{j,i}} = \mathsf{PRF}(k_j, i)$
$\mathcal{L}_x \coloneqq \left\{ \ell_{j,i}^{x_{j,i}} \right\}_{i \in [B]}$
receive $_{\mathsf{M}}(F, \mathsf{sk}, w)$
$Y \leftarrow \mathsf{Ev}\left(F, \mathcal{L}_x || \mathcal{L}_z \right)$
$Y_z || Y_{f(x,z)} \coloneqq Y$
send $_{\mathsf{O}}(Y_z)$
receive $_{\mathsf{O}}(v)$
$t || d_{f(x,z)} = w \oplus v$
if $\mathsf{MACVfy}(\mathsf{sk}, d_{f(x,z)}, t) = 1$
  $f(x, z) \leftarrow \mathsf{De}(d_{f(x,z)}, Y_{f(x,z)})$
  return $f(x, z)$
else return $\perp$

Figure 16: METIS algorithm

Chapter 3

3 A computation over encrypted data in cloud environments for genomic data

3.1 Overview

We propose a system where the data owners have complete control over the genomic data, while reducing the computational costs based on the METIS system employing garbled circuits in an external server to reduce the computational burden on the clients as well as the extensive storage requisites that are required for genomic databases.

The system divides the actuators in the scheme into four different groups, the sequencing center, the server, the data owners and the clients, where in each of the groups different protocols will be employed to encrypt the genomic data to assure that it remains safe during its transfer to the cloud server, when computations are being used on the encrypted data and anonymizing the data on the database to guarantee the security of the identity of any individual reflected in the original data.

Additionally, the data proportioned by the sequencing center is anonymized employing novel synthetic microdata generation techniques to guarantee that no reidentification can be done.

3.2 Description of the protocol

While the protocol follows a similar structure and procedure to the METIS system [41], the protocols and data that it employs have been modified to solve its previous problems while also providing more security with more novel methods.

The system will consist of a service provider system between genetic researchers and data owners, while securing the genetic data in the METIS servers, providing computations on large sets of genetic samples in a remote, secure way and diminishing the storage requisites of the data owners on genomic data, as the extensive data can gather up to

1.5GB of space for a single individual if the full genomic sequence is to be recorded per individual.

For it, it designs four different types of parties: sequencing center, server, data owners and clients, which serves as ways to ease the computation times and storage weights on the clients and data owners, as well as keeping the data secure and accessible, thanks to the employed protocol for data encoding, which leverages the security and integrity properties of garbled circuits, while allowing secure computation of functions.

The protocol starts after the sequencing center finishes the sequencing of the genomes and proceeds to aggregate all the data into a single database whose data will be sent to the cloud servers after it is anonymized employing synthetic microdata generation techniques.

Therefore, after the database has been obtained, in the machines of the sequencing center the algorithm will obtain a new database from the original, constructing new genetic data sets following the principles of population genetics by mixing and hashing initial genetic material over many generations in order to create a genetically disperse future population that shares the same genetic traits in general as the initial population, but holds no direct correspondence to the original individuals by simulating crossovers and mutation events with random pairings.

Finally, the new formed data is encrypted employing a secret key generated from a pseudorandom function, sending both to the data server, where using the key, the genomic sequence can be made use in operations. Said key will be used by the server to form a sequence of random labels of the same length, storing labels in pairs, using a sequence of bits to indicate which of the labels in the pairs are truly random labels and which ones are pseudorandom labels, using the key to easily reconstruct the labels that correspond to the genomic sequence. Also, the genomic data will be split into blocks of biological units, imitating genes, as computing on one block is less expensive computationally, although, if desired, the whole DNA sequence can be represented into one block in order to analyze them fully.

The server will act as a mediator between the parties, involved in all the interactions while storing the encrypted data, but is unable to learn anything from the genome or the input of the client.

Whenever a client wants to evaluate a function on a DNA sequence, it indicates its intention to the data server, which notifies the data owners about the request and generates a garbled circuit with the encoded information, which will be used to evaluate a function by the client, given the permission by the data owner. If it desires the function of the client to be performed on the data, then it will send to the client, automatically, the decoding information towards the client, which can then use to decode the output of the garbled circuit and have their query resolved.

A data owner does not necessarily need to be online at all times to accept or reject petitions of clients to execute functions, as they can define a set of policies that can define whether a client can execute a function or not on their data, hidden to the clients, so the server can take decisions on the owners behalf freely when requests arise, although they can be changed whenever the owner's privacy preferences are desired to be swapped, and the owner can also accept or reject requests personally.

3.2.1 Improvements on the METIS scheme

The main differences that our adaptation of the METIS scheme has over the original consist of the introduction of synthetic microdata generation techniques to anonymize the data obtained from the sequencing center, in order to guarantee that any disclosure of data will not be enough to re-identify another person from the genomic data, protecting their anonymity, thus covering one of the main problems that arise with the use of this information.

Moreover, we replace the traditional Oblivious Transfer techniques that are employed for the data exchange with a novel Oblivious Transfer scheme proposed by *Jianchang Lai et. al.* who developed a **two-round k-out-of-n oblivious transfer scheme with minimal communicational costs** [5], solving the efficiency issues that surge with the traditional OT scheme. The OT scheme that is employed in the traditional METIS protocol is a two-round k-out-of-n oblivious transfer [20], which is expensive, scaling up to lineal costs to the size of the inputs.

This scheme is characterized by employing efficient rounds where the communication costs are reduced to the ideal costs, being the messages from receiver to sender constant, and only three elements being independent from *n* and *k*. The system parameter that they employ is universal, thus being able to be used by any users.

The first round of the protocol commences by sending a token by the receiver, containing the receiver's choice, and a proof information which is used to prove that its choice is not larger than k to the sender. In the second round, the sender responds with encrypted secrets after checking the validity of the received token. Finally, the receiver uses its choice set and secret key to decrypt the ciphertext and only retrieves the secrets whose indexes are in its choice set.

The scheme is as follows:

- **Inputs**:
  System parameter SP. The PKG runs the following *Setup* algorithm: Given a security parameter λ, this algorithm generates a bilinear group
  BG = (G, G_T, e, p) with two generators g, h ∈ G. Then it randomly chooses
  $\alpha \in Z^*_p$ as the system secret key, and computes $g_i = g^{\frac{1}{\propto+i}}$, $h_i = h^{\propto^i}$ for all
  i = 1, 2, …, n. The system parameter SP consists of (BG, g, h, g_1, g_2, …, g_n, h_1,

$h_2, \ldots, h_n$).

S (sender) holds a set of secrets $M = \{m_1, m_2, \ldots, m_n\} \in G_T$.

R (receiver) holds his choice set $G = \{l_1, l_2, \ldots, l_k\}$ C [n].

- **Protocol**:

1. R → S: Given a choice set $G = \{l_1, l_2, \ldots, l_k\}$and the system parameters SP, R picks a random $s \in Z^*_p$ as his secret key sk and uses the *Aggregation* algorithm to compute

$$P(G) = g^{\frac{s}{(\propto+l_1)(\propto+l_2)\cdots(\propto+l_k)}} \quad \Sigma = h^{\frac{(\propto+l_1)(\propto+l_2)\cdots(\propto+l_k)\cdot\alpha^{n-k}}{s}}$$

and sets T = (P(G), Σ, k).

2. S →R: S runs the *Encrypt* algorithm as follows.

Given a set T = (P(G), Σ, k), a set of secrets $M = \{m_1, m_2, \ldots, m_n\}$ and the system parameter SP, it first performs the verification algorithm as: $e(P(G), \Sigma) = e\left(g, h^{\propto^{n-k}}\right)$. If the equation does not hold, it aborts, and otherwise, it accepts |G| ≤ k. Then it picks a random $r \in Z^*_p$ and computes the ciphertext CT for the secrets

$$C_0 = P(G)^r = g^{\frac{s}{(\propto+l_1)(\propto+l_2)\cdots(\propto+l_k)}}$$

together with, for each i = 1, 2, …, n:

$$C_i = e\left(g^{\frac{1}{\propto+1}}, h\right)^r \cdot m_i$$

- **Outputs:**

R runs the decryption algorithm as follows.

Given a cyphertext CT = $(C_0, C_1, \ldots, C_n)$, a choice set $G = \{l_1, l_2, \ldots, l_k\}$, a secret key sk and the system parameter SP, for each i∈G, R computes

$$m_i = C_i \cdot e\left(C_0, h^{\frac{(\propto+l_1)(\propto+l_2)\cdots(\propto+l_k)}{(\propto+i)}}\right)^{-\frac{1}{sk}}$$

A trusted third party called PKG establishes the system by choosing a security parameter λ and a random α as the system secret key, and generates the system parameter SP = (BG, g, h, $g_1, g_2, \ldots, g_n, h_1, h_2, \ldots, h_n$), which is public known.

In the first round, the receiver chooses a random $s \in Z^*_p$ as its secret key and a set $G = \{l_1, l_2, \ldots, l_k\}$. Then, it uses the *Aggregation* algorithm to compute a token $P(G) = g^{\frac{s}{(\propto+l_1)(\propto+l_2)\cdots(\propto+l_k)}}$ and a proof information $\Sigma = h^{\frac{(\propto+l_1)(\propto+l_2)\cdots(\propto+l_k)\cdot\alpha^{n-k}}{s}}$ for its choice set G.

In the second round, upon receiving a request from a receiver, the sender first tests whether $e(P(G), \Sigma) = e\left(g, h^{\propto^{n-k}}\right)$ as a verification algorithm, aborting if it is not equal. Otherwise, S selects a random $r \in Z^*_p$ and computes a ciphertext CT for the secrets as

$C_0 = P(G)^r = g^{\frac{s}{(\propto+l_1)(\propto+l_2)\cdots(\propto+l_k)}}$ together with, for each i = 1, 2, …, n: $C_i = e\left(g^{\frac{1}{\propto+1}}, h\right)^r \cdot m_i$.

Once receiving the encrypted secrets CT from the sender, the receiver computes, for each i ∈ G, $m_i = C_i \cdot e\left(C_0, h^{\frac{(\propto+l_1)(\propto+l_2)\cdots(\propto+l_k)}{(\propto+i)}}\right)^{-\frac{1}{sk}}$. Thus, after the decryption, R only gets k secrets with indexes in G from S.

If i ∈ G, $h^{\frac{(\propto+l_1)(\propto+l_2)\cdots(\propto+l_k)}{(\propto+i)}}$ is computable from the system parameters, so the decryptor can retrieve the encryption (decryption) key $e\left(g^{\frac{1}{\propto+1}}, h\right)^r$ together with $P(G)^r$ and its private key s. If i ∉ G, the value of $h^{\frac{(\propto+l_1)(\propto+l_2)\cdots(\propto+l_k)}{(\propto+i)}}$ cannot be computed, and thus, the decryptor is unable to retrieve the encryption key and get the corresponding secret.

Employing this OT scheme, the communication costs are reduced to the minimum, trading off with a slightly higher computational cost, that allows an efficient secure data exchange for big genomic sequences, as it can be seen when comparing this scheme to other OT schemes (The second implementation done by *Chu and Tzeng* is a protocol similar to the one employed by METIS):

	System Parameter	Messages $(R \to S)$	Messages $(S \to R)$
Mu et al. [27]	$2n\lvert\mathbb{Z}_p\rvert$	$2n$	$2n$
Zhang and Wang [34]	$\lvert\mathbb{G}\rvert$	$k+3$	$2n$
Chu and Tzeng [8]	$\lvert\mathbb{G}\rvert$	k	$n+k+1$
Chu and Tzeng [7]	$\lvert\mathbb{G}\rvert$	k	$n+k$
Guo et al. [14]	$(n^2+3n+3)\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$	4	$3n$
Guo et al. [15]	$(n+1)\lvert\mathbb{G}\rvert$	3	$2n$
Jianchang Lai et. al.	$(2n+2)\lvert\mathbb{G}\rvert$	3	$n+1$

Figure 17: Comparison of two-round OT_n^k in terms of communication cost.

Comparing the computation costs of the two algorithms, the implementation employed by METIS has a computation complexity as follows: R computes 3k and S computes kn modular exponentiations, whereas the new OT scheme that we propose proposes a tradeoff, reducing the communication overheads while adding more computation, as it

has to encrypt and send all secrets in order to reduce for the sender the communication times, thus obtaining computation costs for the R equal to 3k, and for S computes (k+3)n exponentiation operations.

Thanks to this tradeoff, the communication is to be as quick as possible as to haste process of the modified METIS system as much as possible, while not increasing the computation costs in an excessive manner, thus providing to be a better alternative than the original one.

Another change that has been implemented on the original METIS scheme consists of the introduction of the possibility to execute **secure count queries** on the encrypted genomic data [25]. Many cryptographical works ignore the potential of the study of SNP sequences, as it is mainly a biological term, but a single change of a SNP can correlate to many different illnesses, characteristics, diagnoses and so, thus implementing a way to make SNP queries on the data is of high importance for future research.

In count query operations the main aim is to know how many records in the database match a given query predicate (i.e., a certain combination of genotype and phenotype values).

For this, we implement a system based on the framework proposed by *Hasan, M.Z. et. al.* The architecture of the system that they propose divide the participants into four groups: Data Owners, Certified Institution (CI), Cloud Server (CS) and Researchers. Each entity is responsible for performing different specific tasks to make the overall system secure and functional. The Data Owners, Cloud Server and Researchers correspond to the roles we already specified before of Data Owner, Server and Client roles, while the Certified Institution is the one that gets the data into a database, being a trusted entity, like the National Institute of Health (NIH). With the data, it builds an encrypted searchable version of the aggregate shared data and sends it to the CS. The search operation is basically performed on an encrypted index tree.

In our proposed system, the sequencing center will be the one in charge of building a prefix tree from the dataset that contains all the records from aggregate shared data and sends the encrypted version of the tree to the server, where the server will manage the keys used for encryption and decryption [40].

The scheme starts after the data has been obtained and encrypted into labels for normal processing in our METIS-like scheme.

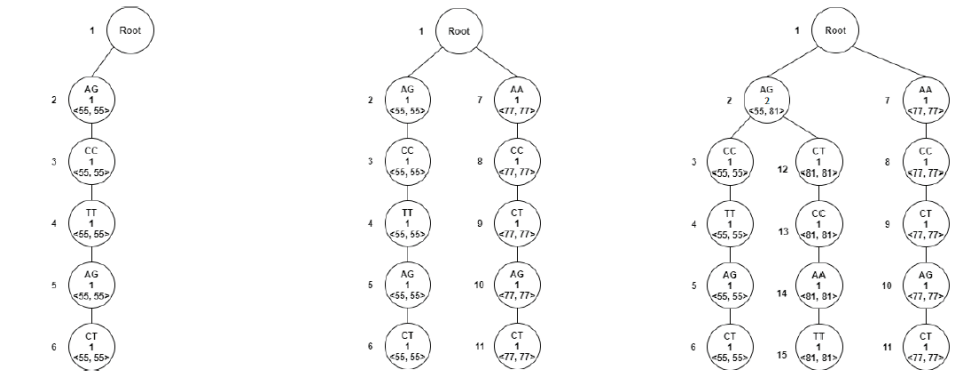
The sequencing center then generates a prefix tree T, using as attributes the SNPs, phenotype and age to distinguish in the use of the SNPs. It encodes each SNP from 1 to 16, as there are 16 possible SNP sequences, for each record. Each node in the prefix tree contains:

- *sid*: It represents the unique identifier for a SNP.
- *Val*: It represents the SNP's value (AA, AG, CC, ..., TT), which is encoded as {1, 2, ..., 16}.
- *Count*: It represents the total number of occurrences of a SNP in a particular position of the prefix tree.
- *Phenotype*: Every node i of T consists of a Bloom filter Bi to hold the phenotype information.
- *List*: The children list of a node.
- *Additional information*: Numeric information such as age range is also saved on the nodes. Each node contains of *age.low* and *age.high* that represents the age range of the current node and its children.

A node is represented as s δ (sid, val, count, phenotype, list, age.low, age.high). The prefix tree T is generated following this algorithm:

At first, the tree T only contains the root node. For each record in D, we start from the root node and create new nodes in T. We represent a record as $d_i^j \in D$ where *i* and *j* represents the particular record and the particular column, respectively. For each d_i^j, the first column d_i^1 represents the root node's child. The second column, d_i^2, is the child of the node d_i^1, and it iterates this procedure to create the rest of the tree.

Thus, level 1 represents the first column's data, level 2 represents the second column's data, and so forth, as seen in the *Figure 18*.



(a) First record insertion. (b) Second record insertion. (c) Third record insertion.

Figure 18: Prefix tree generation states.

When needing to insert a new record, d_2, we first check each of the corresponding level's columns of T whether the current record has been inserted in T before or not. If we don't find any, we create a new node to store d_2^j. If it exists on the tree, then we increment the count value.

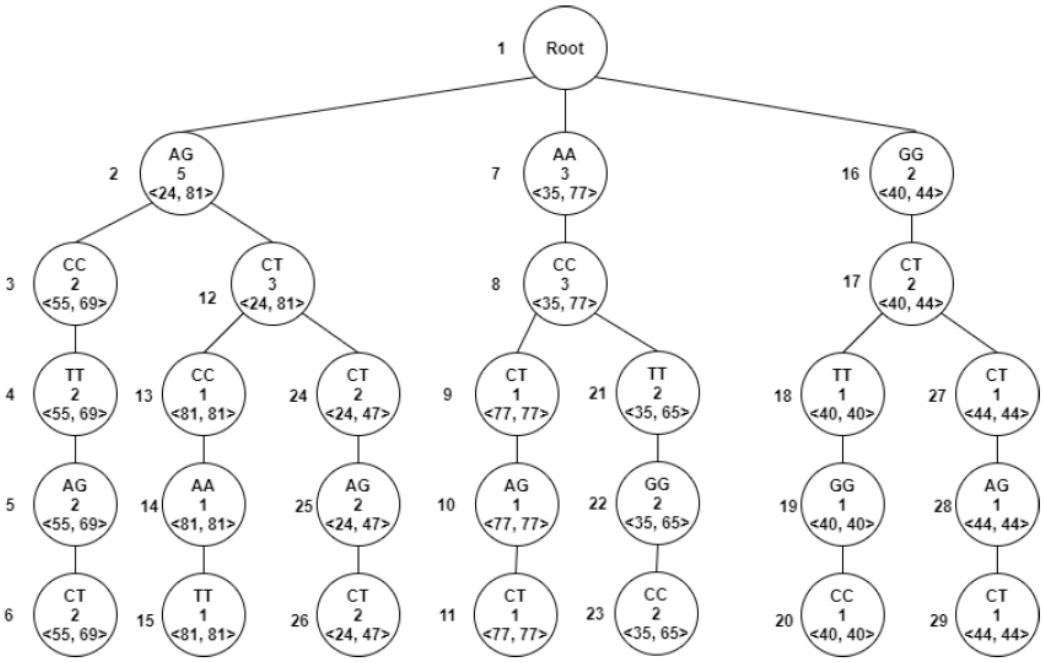


Figure 19: Example of fully constructed prefix tree.

The process of constructing the prefix tree is O(nm), where n is the total number of records in the database, and m is the total number of SNPs per record. This algorithm is to be executed at the same time, in parallel to the construction of the anonymized database for the METIS protocol.

Then the tree is encrypted, generating two keys employing the *Paillier Cryptosystem*, the *pk* and the *sk*, and encrypts the nodes of T using the public key *pk*, in an analogous way to the METIS system.

Each node in the tree becomes $(\sigma(sid, \varepsilon_{pk}(val), \varepsilon_{pk}(count), \varepsilon_{pk}(\beta_i), \varepsilon_{pk}(age.low), \varepsilon_{pk}(age.high), list)$ after the encryption. This encrypted tree is depicted as $\tilde{T}$.

Finally, both the tree (*pk*, $\tilde{T}$)and the database are sent to the servers for the queries of the clients.

To execute a encrypt query on the tree $\tilde{T}$, the query first must be constructed according to the encryption employed for the tree. The researchers encrypt a query q as $\Phi\ (\varepsilon_{pk}(q))$, employing the public key that is shown by the server.

The age fields of the query are not needed to be encrypted, as those are sent directly to the garbled circuit to ease the query construction.

```
SELECT COUNT(*) FROM Sequences
    WHERE SNP2='+a=#?h' AND SNP4='z@0ux&*'
          AND Diagnoses='1011011....11'
            AND Age between 35 to 65
```

Figure 20: Example query for a prefix tree.

The encrypted query Φ is sent to the server to find the number of records, which matches the SNPs, phenotype, and age range in the query predicate.

To search numeric information in prefix tree, we need to check the age range of the query whether it lies between the node.age.low and node.age.high, which are easily done employing garbled circuits, allowing us to compare both ages to the ones sent in the query in their encrypted forms.

Chapter 4

4 Implementation of the synthetic genomic data generation algorithm

4.1 Overview

We propose an algorithm that takes advantage of the properties of the phenotypic proportions that apply to the genome sequences, supporting the privacy of the genomic data before it's sent from the sequencing center.

This algorithm takes hold on the ideas for dissemination of genetic data by Anna Oganian, who discusses novel techniques to solve the privacy problems in the dissemination of genetic data by employing synthetic data. We modify the obtained genomic by simulating the recombination events in the mitosis, recombining all the individual's data as to produce new sequences.

One per iteration, each individual swaps its second half of their genotype with another random individual, with individuals that did not were swapped in the iteration. After the genotypes are swapped, each individual is subjected to possible mutations, simulating the possible errors that can happen during the biological processes, which lead to more distance between the original genotype and the synthetic one. The mutations considered are the most common mutations that can happen, base substitutions, where a nitrogenated base is replaced with another one.

There are two kinds of mutations that result in a base substitution, where the bases can be catalogized into two categories, purines (being those the adenine and guanine) and pyrimidines (being the thymidine and guanosine) [17]:

- **Transition**: When a purine is substituted with another purine or when a pyrimidine is substituted with another pyrimidine.
- **Transversion**: When a purine is substituted for a pyrimidine, or a pyrimidine replaces a purine.

It is worth noting that, although there are two possible transversions but only one possible transition per base, transition mutations are more likely than transversions because substituting a single ring structure for another single ring structure is more likely than substituting a double ring for a single ring (due to the structural differences of pyrimidines

and purines). Also, transitions are less likely to result in amino acid substitutions (due to wobble base pair) and are therefore more likely to persist as "silent substitutions" in populations as single nucleotide polymorphisms (SNPs).

These combinations do not break the data that they provide, as the principles of population genetics indicate that the phenotypic proportions of the main characteristics and SNPs of the individuals would still be present in the population in the same proportions as they were in the original set, allowing to create a genetically disperse future population that shares the same genetic traits in general as the initial population, but holds no direct correspondence, thanks to the recombinations and crossovers of the genetic material, as can be seen in *Figure 8*.

Finally, we can employ other genomic tools to verify manually that the phenotypes are not changed too far from the original data, testing different possible chances of mutations, checking how much we can modify our data without straining too far from the original information, or without breaking the sense of the genomic sequences.

4.2 Description of the algorithm

We will now describe how the simple genomic algorithm operates, based on the system employed by the software '*Haploview'*, developed by Broad institute at MIT.

The input of the algorithm will be based on the files employed for the software, using the generic format for displaying genomic data, the Linkage format. This format is based on the LIPED format, is a format used to store data for software capable of analyzing marker genotypes, and specifies a staple to store genomic data, saving the references to the individuals without exposing them directly.

It employs two different files, a '.ped' file (Linkage Pedigree) [6] which stores the family IDs, the individual IDs to reference the genotype' progeny, the father and mother's IDs, the sex, the affection status, employed for association tests, and finally the marker genotypes, displaying the nitrogenated bases of the genotype, and a 0 if the data is missing. The genotype employed is introduced in haploid format (major and minor alleles), so we only employ one-chained DNA sequences.

	A	B	C	D	E	F	G	H	I	J	K
1	1	Ind_1	0	0	1	0	G	G	G	G	G
2	1	Ind_2	0	0	1	0	G	G	G	G	G
3	1	Ind_3	0	0	1	0	C	C	A	A	A
4	1	Ind_4	0	0	1	0	C	C	A	A	A
5	1	Ind_5	0	0	1	0	C	C	A	A	A
6	1	Ind_6	0	0	1	0	G	G	G	G	G
7	1	Ind_7	0	0	1	0	C	C	A	A	G
8	1	Ind_8	0	0	1	0	G	G	G	G	G
9	1	Ind_9	0	0	1	0	C	C	A	A	A
10	1	Ind_10	0	0	1	0	G	G	G	G	G
11	1	Ind_11	0	0	1	0	G	G	G	G	G
12	1	Ind_12	0	0	1	0	G	G	G	G	G
13	1	Ind_13	0	0	1	0	G	G	G	G	G
14	1	Ind_14	0	0	1	0	G	G	G	G	G
15	1	Ind_15	0	0	1	0	G	G	G	G	G
16	1	Ind_16	0	0	1	0	G	G	G	G	G
17	1	Ind_17	0	0	1	0	G	G	G	G	G
18	1	Ind_18	0	0	1	0	G	G	G	G	G
19	1	Ind_19	0	0	1	0	C	C	A	A	G

Figure 21: Reference of .ped file

The second file that this format employs is the '.info' file, consisting of two columns, which indicate the marker information of the genomic sequences, allowing to extract the phenotype out of the genotype. The two attributes that it stores are the marker name, which can have a stablished name, or can be specified manually, if it's a known marker or not, and the physical position of it in the genomic sequence.

	A	B
1	marker_1	30789
2	marker_2	30838
3	marker_3	31461
4	marker_4	32475
5	marker_5	140209
6	marker_6	515015
7	marker_7	596470
8	marker_8	608683
9	marker_9	662422
10	marker_10	695932

Figure 22: Reference of .info file.

The algorithm operates using the python language, employing the pandas, math, numpy and random libraries in order to operate the files, compute the chances for the mutation and manage the vectors that contain the genotype of an individual.

It proceeds by extracting the genotypic sequence from the .ped file information into a separate dataframe, which will be the one that will be modified.

After that, the program will iterate for the specified number of generations that we want to simulate.

In each generation, we first randomly choose pairs of two individuals and swap their latter half of their genotypes, until no individuals remain without recombining their genotypes, simulating a haploid recombination of the DNA sequences, simulating the strand assimilation process of the recombination, in which two strands merge together swapping their genome.

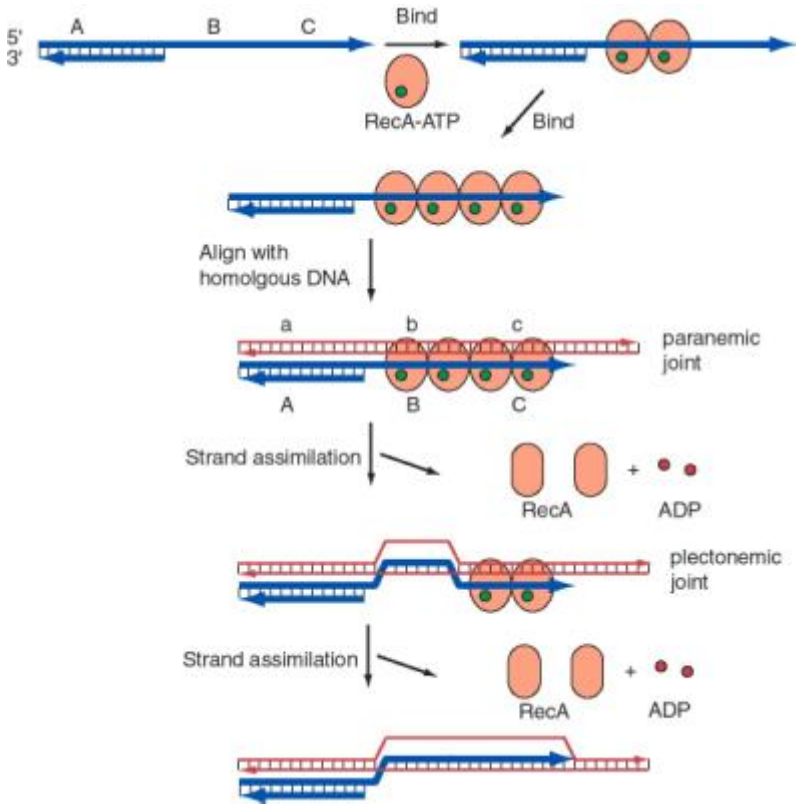


Figure 23: Monoantennary DNA assimilation process.

Then, immediately after we check per individual if a mutation happens in each nitrogenated base, following the possibility indices specified at the beginning for the transition and the transversion mutations (we stablish the transversion mutation chance to be the half value of the transition mutation chance, as transversion base substitution mutations are far more difficult to happen since these kinds of mutations have to modify the nitrogenated base by modifying the ring structure to be a double nitrogenated ring instead of a single one (pyrimidine to a purine) or vice versa).

These mutations are done after the recombinations and not at the same time as to not modify the genomic data during the recombination, which might lead to some strands to be mutated more than once, unlike in the real recombination events, where all these events happen during the same process.

With this, the generation step is completed for new iterations.

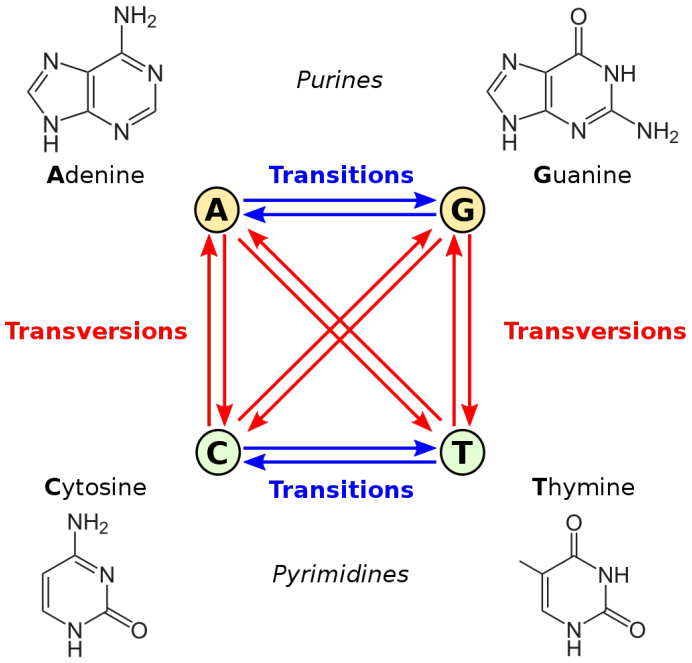


Figure 24: Base substitution mutations: Transitions and transversions.

```
for i in range(num_generations):
    isModified = np.full(len(df_genome.index), False)
    for index, row in df_genome.iterrows():
        if (not isModified[index]):
            isModified[index] = True
            notModifiedIndividuals = np.where(~isModified)[0]
            if (len(notModifiedIndividuals) != 0):
                combIndividual = np.random.choice(np.asarray(notModifiedIndividuals))
                temp = df_genome.iloc[combIndividual, genotypeMid:].copy()
                df_genome.iloc[combIndividual, genotypeMid:] = df_genome.iloc[index, genotypeMid:].copy()
                df_genome.iloc[index, genotypeMid:] = temp.copy()
                isModified[combIndividual] = True
                print("Combined",index,"with",combIndividual)
    #After the combinations have been done, one by one is checked for mutations in their genome
    for index, row in df_genome.iterrows():
        for col in range(genotypeLength):
            #Randomly choose the possibility of a transition or a transversion.
            #Mutation type = 1 if transversion, 0 if transition
            mutationType = 1 if random.uniform(0,1) > 0.5 else 0
            mutationChance = random.uniform(0,1)
            if (mutationType == 0):
                if (mutationChance <= transitionMutationRate):
                    original = df_genome.iloc[index,col]
                    if (df_genome.iloc[index,col] == 'C'):
                        df_genome.iloc[index,col] = 'A'
                    elif (df_genome.iloc[index,col] == 'A'):
                        df_genome.iloc[index,col] = 'C'
                    elif (df_genome.iloc[index,col] == 'T'):
                        df_genome.iloc[index,col] = 'G'
                    elif (df_genome.iloc[index,col] == 'G'):
                        df_genome.iloc[index,col] = 'T'
                    print("Transition mutation (chance:", mutationChance,") in Row:", index, ", Col:", col, "Changed base from ",
            else:
                if (mutationChance <= transversionMutationRate):
                    original = df_genome.iloc[index,col]
                    if (df_genome.iloc[index,col] == 'C'):
                        df_genome.iloc[index,col] = 'G' if random.uniform(0,1) > 0.5 else 'T'
                    elif (df_genome.iloc[index,col] == 'A'):
                        df_genome.iloc[index,col] = 'G' if random.uniform(0,1) > 0.5 else 'T'
                    elif (df_genome.iloc[index,col] == 'T'):
                        df_genome.iloc[index,col] = 'A' if random.uniform(0,1) > 0.5 else 'C'
                    elif (df_genome.iloc[index,col] == 'G'):
                        df_genome.iloc[index,col] = 'A' if random.uniform(0,1) > 0.5 else 'C'
                    print("Translocation mutation (chance:", mutationChance,") in Row:", index, ", Col:", col, "Changed base from
```

Figure 25: Data anonymization iterating by desired amount of generations

Finally, after all the generations have been completed, the genotype is joined back with the genotype information, and exported into a .csv file, which can be used to make a .ped file for the analysis of the data using external software, which in our case, is Haploview.

4.3 Analysis of the results

4.3.1 Haploview software

The Haploview software [19] is a Java based open-source software, developed by Broad institute at MIT for haplotype analysis and LD visualization. It is designed to simplify and expedite the process of haplotype analysis by providing a common interface to several tasks relating to such analyses, providing investigators the possibility to perform wide range of analyses such as LD & haplotype block analysis, haplotype frequency estimation,

association analysis, and visualization and plotting LD and haplotypes parting from two files following the Linkage format.

After loading a file, Haploview shows some basic data quality checks for the markers. Markers are filtered out based on some default criteria which can be adjusted as necessary, showing the information in different columns for each specified marker in the .info file.

These columns indicate the following information:

- # is the marker number.
- Name is the marker ID specified (only if an info file is loaded).
- Position is the marker position specified (only if an info file is loaded).
- ObsHET is the marker's observed heterozygosity (which refers to the percentage of gene loci that are heterozygous in an average individual of a given population.)
- PredHET is the marker's predicted heterozygosity (i.e. 2*MAF*(1-MAF)).
- HWpval is the Hardy-Weinberg equilibrium p value, which is the probability that its deviation from H-W equilibrium could be explained by chance.
- %Geno is the percentage of non-missing genotypes for this marker.
- FamTrio is the number of fully genotyped family trios for this marker (0 for datasets with unrelated individuals).
- MendErr is the number of observed Mendelian inheritance errors (0 for datasets with unrelated individuals).
- MAF is the minor allele frequency (using founders only) for this marker.
- Alleles are the major and minor alleles for this marker.
- Rating is checked if the marker passes all the tests and unchecked if it fails one or more tests (highlighted in red).

#	Name	Position	ObsHET	PredHET	HWpval	%Geno	FamTrio	MendErr	MAF	Alleles	Rating
1	marker_1	30789	0.001	0.424	1.5622E-196	99.7	0	0	0.305	G:C	
2	marker_2	30838	0.003	0.41	2.4257E-184	97.3	0	0	0.287	G:A	
3	marker_3	31461	0.001	0.226	3.0569E-122	99.6	0	0	0.13	G:A	
4	marker_4	32475	0.001	0.424	1.0865E-196	99.9	0	0	0.305	G:A	
5	marker_5	140209	0.003	0.266	1.4449E-136	99.9	0	0	0.158	A:C	
6	marker_6	515015	0.003	0.274	3.5145E-138	98.9	0	0	0.164	A:G	
7	marker_7	596470	0.003	0.284	1.1488E-143	100.0	0	0	0.171	A:C	
8	marker_8	608683	0.001	0.15	9.722E-89	99.5	0	0	0.081	C:A	
9	marker_9	662422	0.001	0.12	2.1322E-74	99.5	0	0	0.064	G:A	
10	marker_10	695932	0.004	0.451	5.3136E-201	99.9	0	0	0.344	A:G	
11	marker_11	764133	0.001	0.157	1.0696E-91	98.9	0	0	0.086	G:A	
12	marker_12	832195	0.004	0.257	5.4028E-130	99.3	0	0	0.152	A:G	
13	marker_13	840314	0.004	0.418	3.9167E-190	100.0	0	0	0.298	A:G	
14	marker_14	1047561	0.008	0.408	4.3742E-180	100.0	0	0	0.285	C:G	
15	marker_15	1093747	0.001	0.228	5.2938E-123	99.5	0	0	0.131	C:A	
16	marker_16	1219776	0.001	0.206	1.4421E-113	99.2	0	0	0.117	C:A	
17	marker_17	1222568	0.0	0.141	1.0978E-87	99.7	0	0	0.077	G:A	
18	marker_18	1255163	0.014	0.396	1.2118E-166	99.1	0	0	0.272	A:C	
19	marker_19	1265191	0.003	0.326	1.688E-159	100.0	0	0	0.205	C:A	
20	marker_20	1461490	0.004	0.397	1.9111E-182	99.9	0	0	0.273	A:G	
21	marker_21	1483008	0.003	0.236	1.7511E-124	100.0	0	0	0.137	G:A	
22	marker_22	1483058	0.001	0.107	1.531E-68	99.9	0	0	0.057	A:G	
23	marker_23	1484632	0.001	0.189	1.6374E-106	99.6	0	0	0.106	G:A	
24	marker_24	1485494	0.003	0.352	6.0687E-166	98.3	0	0	0.228	G:A	
25	marker_25	1870690	0.032	0.228	1.724E-84	99.1	0	0	0.131	G:A	
26	marker_26	1881207	0.005	0.466	6.5164E-204	100.0	0	0	0.369	G:A	
27	marker_27	1883270	0.034	0.291	1.3146E-106	98.9	0	0	0.177	A:G	
28	marker_28	1986674	0.007	0.45	7.2937E-196	99.7	0	0	0.342	A:G	
29	marker_29	2114294	0.008	0.468	3.7069E-200	99.9	0	0	0.374	A:G	

Figure 26: Marker information checks in Haploview.

We can also obtain results following Linkage Disequilibrium Displays (LD Displays).

The Linkage Disequilibrium [9], [10] is the non-random association of alleles at different loci in a given population. Loci are said to be in linkage disequilibrium when the frequency of association of their different alleles is higher or lower than what would be expected if the loci were independent and associated randomly.

This means, that the LD represents how probable are different alleles to be associated together more frequently. The LD is represented in a graphic display, showing the LD values for two different markers as blocks, representing the LD value of two markers as the block that connects the two diagonally. If no value is presented, it means that it has a LD value of a hundred, meaning that the two markers are always associated together.

Recombination interacts in a complex way with selection, mutation and genetic drift to determine levels of LD. As a consequence, local and genome-wide patterns of LD can provide insight into patterns of natural selection and the past history of population growth and dispersal.

In humans and other model organisms, LD between marker alleles and traits of interest allow fine-scale gene mapping [8]. Unusually high local LD can indicate an allele that has recently increased to high frequency under strong selection [38].

The program, once groups of markers are selected, the program generates haplotypes and their population frequencies, partitioning the region into different segments with strong LD [15].

This display shows lines to indicate transitions from one block to the next with frequency corresponding to the thickness of the line and also presents Hedrick's multiallelic D, which represents the overall degree of LD between two blocks, treating each haplotype within a block as an 'allele' of that region [13].

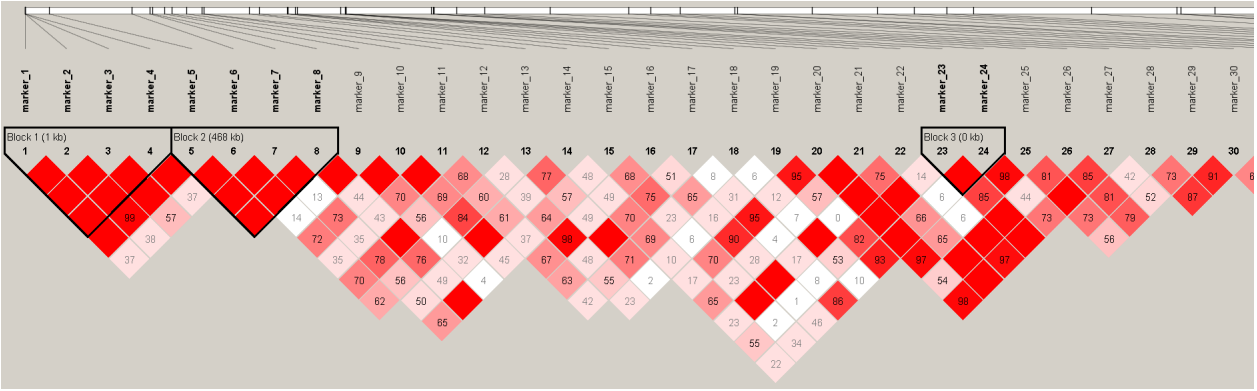


Figure 27: Haploview LD display with recombination rate plotted.

Each box displays the LD statistics for the comparison of any two markers as its Hedrick's multiallelic D value, so we can how the different markers associate in a non-random way [14].

These values can also be represented with their haplotypes, indicating how they relate with each other in what's known as a Haplotype map. It simplifies the individual DNA markers into subsets of alleles that lie close to a single chromosome frequently inherited, thus displaying the relationship between different marker compositions, and how usual they are displayed one another with their current aspect.

Each column represents a single SNP that forms part of the certain block, while each row that appears in the block indicates a possible haplotype representation [38].

The lines that connect the different blocks indicate the locations where more than 1% (10% if the line is thick) of all the chromosomes are observed to transition from one common haplotype to a different one. The number at the right of each block indicates the population frequencies, which indicate the percentage of observed chromosomes that match one of the common patterns exactly, the most common crossings from one block to the next [39].

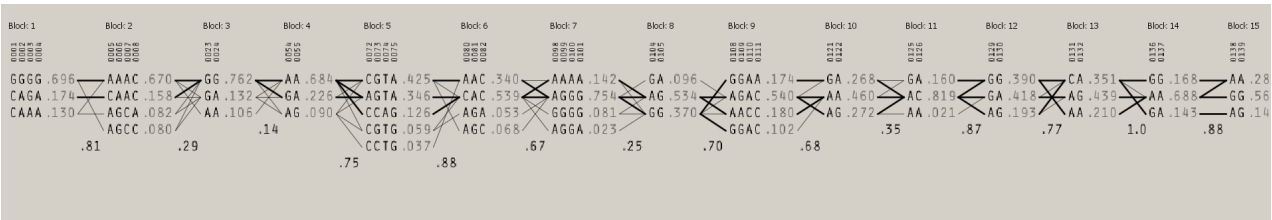


Figure 28: Haploview Haplotype display with transition rates and pattern matches rate.

We can employ these tools that Haploview provides investigators to examine how effective our algorithm is, comparing if the haplotype structure is modified by our changes, and testing how the number of generations and the mutation rates affect these results.

4.3.2 Algorithm results

For the results to be easier to understand, the main factor that will be observed are the differences in how the haplotypes are represented in the different displays, and not analyzing the results showed by the LD display and the haplotype display. If our algorithm has affected the haplotype significantly, then the graphics will show different structures and values, which would indicate that the haplotypes are no longer identical, and therefore, useful for the investigation, and elsewise, were the results showing similar representations, then the algorithm would be successful at anonymizing the original donor's genotypes.

Our algorithm will prove appropriate if the changes that we do to our genotypes do not reflect considerable differences on the phenotypes, as previously stated by the principles of population genetics.

The test dataset is obtained from an investigation done by Daly etal. (2001), who studied the Chron disease applying an LD approach, where the datasets were gathered from the National Library of Medicine / National Center for Biotechnology Information [7][11], providing free collections of genomic data.

It sequences the haploid genotypes of hundreds of individuals affected with the Chron disease and the information of the important SNP markers that want to be studied for the disease, provided in the .ped and .info files needed for the analysis.

1	Ind_1	0	0	1	0	G	G	G	G	G	G	G	G	A	A	A	A
1	Ind_2	0	0	1	0	G	G	G	G	G	G	G	G	A	A	G	G
1	Ind_3	0	0	1	0	C	C	A	A	A	A	A	A	A	A	G	G
1	Ind_4	0	0	1	0	C	C	A	A	A	A	A	A	A	A	A	A
1	Ind_5	0	0	1	0	C	C	A	A	A	A	A	A	A	A	G	G
1	Ind_6	0	0	1	0	G	G	G	G	G	G	G	G	A	A	G	G
1	Ind_7	0	0	1	0	C	C	A	A	G	G	A	A	C	C	A	A
1	Ind_8	0	0	1	0	G	G	G	G	G	G	G	G	A	A	A	A
1	Ind_9	0	0	1	0	C	C	A	A	A	A	A	A	A	A	A	A
1	Ind_10	0	0	1	0	G	G	G	G	G	G	G	G	A	A	A	A
1	Ind_11	0	0	1	0	G	G	G	G	G	G	G	G	A	A	A	A
1	Ind_12	0	0	1	0	G	G	G	G	G	G	G	G	A	A	A	A
1	Ind_13	0	0	1	0	G	G	G	G	G	G	G	G	A	A	A	A
1	Ind_14	0	0	1	0	G	G	G	G	G	G	G	G	A	A	A	A
1	Ind_15	0	0	1	0	G	G	G	G	G	G	G	G	A	A	A	A
1	Ind_16	0	0	1	0	G	G	G	G	G	G	G	G	A	A	A	A
1	Ind_17	0	0	1	0	G	G	G	G	G	G	G	G	A	A	A	A
1	Ind_18	0	0	1	0	G	G	G	G	G	G	G	G	A	A	A	A
1	Ind_19	0	0	1	0	C	C	A	A	G	G	A	A	A	A	G	G
1	Ind_20	0	0	1	0	G	G	G	G	G	G	G	G	A	A	A	A
1	Ind_21	0	0	1	0	C	C	A	A	G	G	A	A	A	A	G	G
1	Ind_22	0	0	1	0	C	C	A	A	G	G	A	A	A	A	G	G
1	Ind_23	0	0	1	0	C	C	A	A	G	G	A	A	A	A	G	G
1	Ind_24	0	0	1	0	G	G	G	G	G	G	G	G	A	A	A	A
1	Ind_25	0	0	1	0	G	G	G	G	G	G	G	G	A	A	A	A
1	Ind_26	0	0	1	0	C	C	A	A	G	G	A	A	A	A	G	G
1	Ind_27	0	0	1	0	C	C	A	A	G	G	A	A	A	A	G	G

Figure 29: Chron disease patients' data

As to keep the results as simple as possible, we will not go into details of the biological results, but instead, checking if the data obtained from Haploview on the original data is

similar or correspondent to the synthetic data that we obtain, thus revealing if modifying the genotypes, we can produce considerable variations in the population's overall haplotype or not.

The original data provides the following results:

#	Name	Position	ObsHET	PredHET	HWpval	%Geno	FamTrio	MendErr	MAF	Alleles
1	marker_1	30789	0.001	0.424	1.5622E-196	99.9	0	0	0.305	G:C
2	marker_2	30838	0.003	0.41	2.4257E-184	97.5	0	0	0.287	G:A
3	marker_3	31461	0.001	0.226	3.5102E-122	99.6	0	0	0.13	G:A
4	marker_4	32475	0.001	0.424	1.5622E-196	99.9	0	0	0.305	G:A
5	marker_5	140209	0.003	0.266	1.4449E-136	100.0	0	0	0.158	A:C
6	marker_6	515015	0.003	0.274	4.1964E-138	98.9	0	0	0.164	A:G
7	marker_7	596470	0.003	0.284	1.3843E-143	100.0	0	0	0.172	A:C
8	marker_8	608683	0.001	0.15	9.722E-89	99.6	0	0	0.081	C:A
9	marker_9	662422	0.001	0.12	2.2763E-74	99.5	0	0	0.064	G:A
10	marker_10	695932	0.004	0.452	8.0759E-201	99.9	0	0	0.344	A:G
11	marker_11	764133	0.001	0.157	1.0696E-91	99.1	0	0	0.086	G:A
12	marker_12	832195	0.004	0.258	6.3539E-130	99.3	0	0	0.152	A:G
13	marker_13	840314	0.004	0.419	5.5638E-190	100.0	0	0	0.298	A:G
14	marker_14	1047561	0.008	0.408	6.0863E-180	100.0	0	0	0.286	C:G
15	marker_15	1093747	0.001	0.228	5.2938E-123	99.6	0	0	0.131	C:A
16	marker_16	1219776	0.001	0.206	1.4421E-113	99.3	0	0	0.117	C:A
17	marker_17	1222568	0.0	0.141	1.0978E-87	99.9	0	0	0.077	G:A
18	marker_18	1255163	0.014	0.396	1.2118E-166	99.2	0	0	0.272	A:C
19	marker_19	1265191	0.003	0.326	2.1196E-159	100.0	0	0	0.205	C:A
20	marker_20	1461490	0.004	0.397	2.6214E-182	99.9	0	0	0.273	A:G
21	marker_21	1483008	0.003	0.236	2.025E-124	100.0	0	0	0.137	G:A
22	marker_22	1483058	0.001	0.107	1.531E-68	100.0	0	0	0.057	A:G
23	marker_23	1484632	0.001	0.189	1.6374E-106	99.7	0	0	0.106	G:A
24	marker_24	1485494	0.003	0.352	6.0687E-166	98.4	0	0	0.228	G:A
25	marker_25	1870690	0.032	0.228	1.9473E-84	99.1	0	0	0.131	G:A
26	marker_26	1881207	0.005	0.466	1.0295E-203	100.0	0	0	0.37	G:A
27	marker_27	1883270	0.034	0.291	1.3146E-106	99.1	0	0	0.177	A:G
28	marker_28	1986674	0.007	0.45	7.2937E-196	99.9	0	0	0.342	A:G
29	marker_29	2114294	0.008	0.468	5.8859E-200	99.9	0	0	0.374	A:G

Figure 30: Marker information for Chron patients

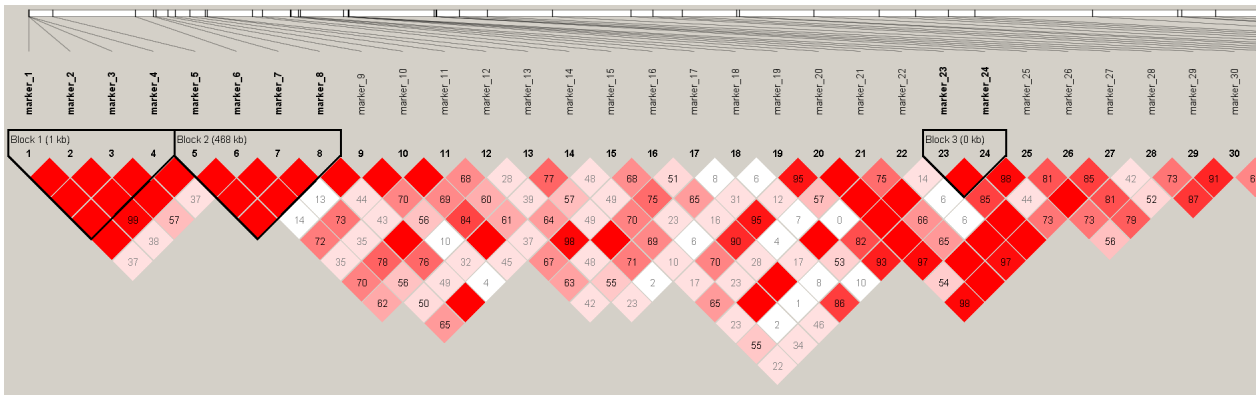


Figure 31: LD display for Chron patients

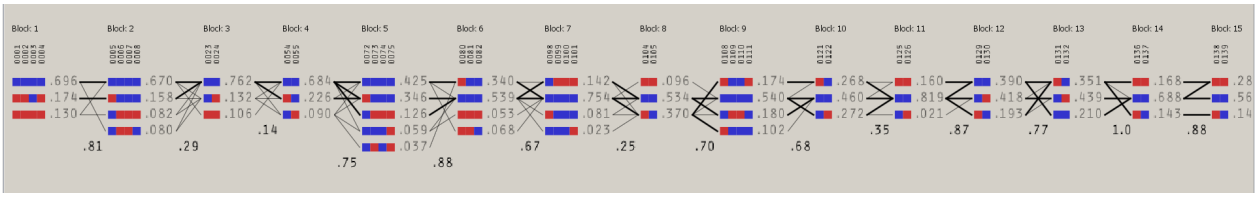
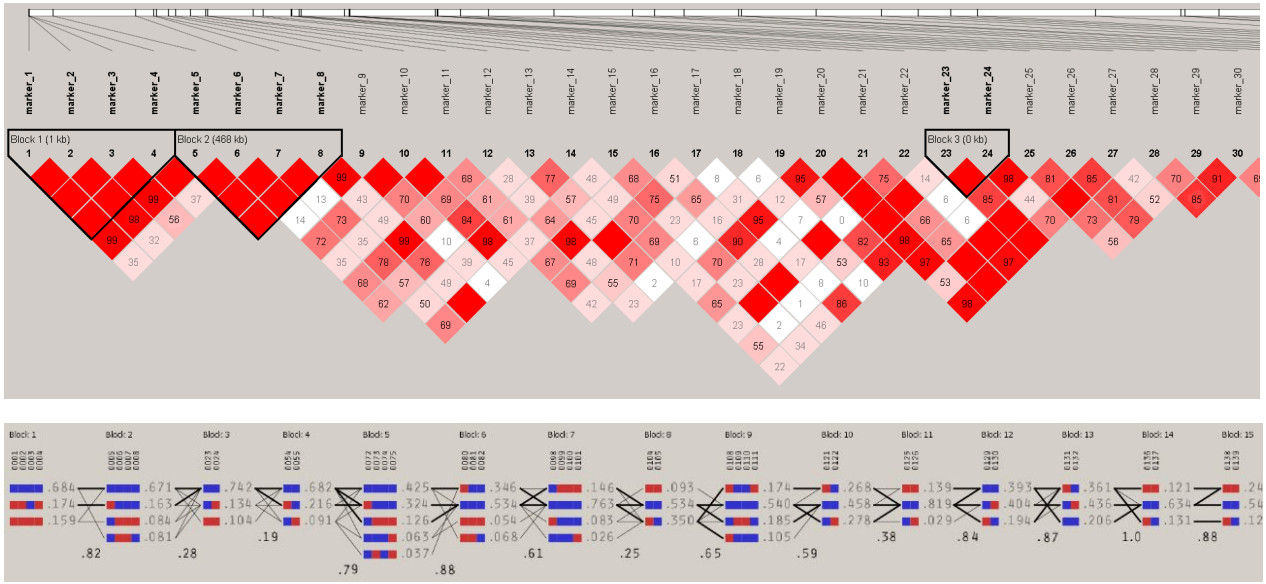


Figure 32: Haplotype display for Chron patients

We will compare now these results to the data obtained from employing the algorithm with different amounts of generations and mutation rates, checking if the population data is preserved, and how do those parameters affect execution.

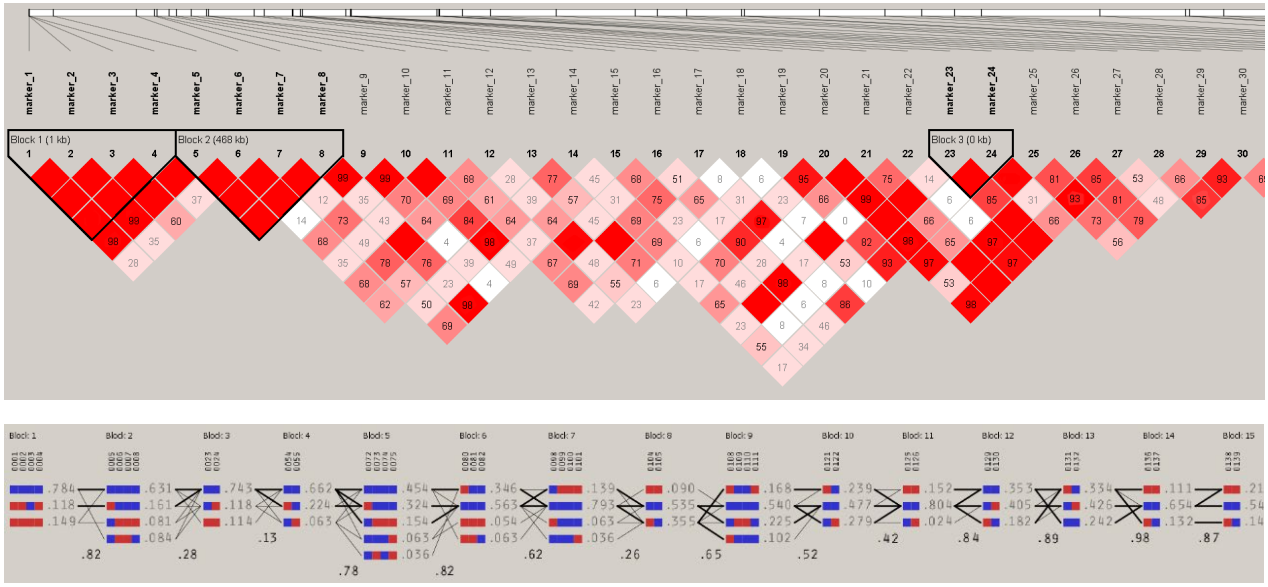
Number of generations: 2 | No mutations

#	Name	Position	ObsHET	PredHET	HWpval	%Geno	FamTrio	MendErr	MAF	Alleles
1	marker_1	30789	0.002	0.454	1.5622E-196	99.5	0	0	0.304	G:C
2	marker_2	30838	0.003	0.41	2.4257E-186	97.5	0	0	0.284	G:A
3	marker_3	31461	0.001	0.226	3.6969E-122	99.6	0	0	0.134	G:A
4	marker_4	32475	0.002	0.454	1.6965E-186	99.9	0	0	0.305	G:A
5	marker_5	140209	0.003	0.266	1.4869E-136	99.4	0	0	0.158	A:C
6	marker_6	515015	0.003	0.274	3.5145E-138	98.9	0	0	0.160	A:G
7	marker_7	596470	0.004	0.284	1.1488E-169	100.0	0	0	0.151	A:C
8	marker_8	608683	0.002	0.15	3.722E-89	99.4	0	0	0.081	C:A
9	marker_9	662422	0.001	0.15	2.1322E-74	99.4	0	0	0.064	G:A
10	marker_10	695932	0.004	0.451	5.3186E-201	99.9	0	0	0.344	A:G
11	marker_11	764133	0.001	0.158	1.0686E-91	99.9	0	0	0.086	G:A
12	marker_12	832195	0.004	0.257	3.8656E-130	99.3	0	0	0.152	A:G
13	marker_13	840314	0.004	0.418	3.9167E-169	100.0	0	0	0.298	A:G
14	marker_14	1047561	0.008	0.408	4.3742E-180	100.0	0	0	0.285	C:G
15	marker_15	1093747	0.001	0.228	5.2938E-123	99.5	0	0	0.131	C:A
16	marker_16	1219776	0.001	0.208	1.4421E-113	99.2	0	0	0.157	C:A
17	marker_17	1222568	0.0	0.141	3.5678E-87	99.9	0	0	0.077	G:A
18	marker_18	1255163	0.014	0.396	1.2118E-169	99.1	0	0	0.272	A:C
19	marker_19	1265191	0.003	0.326	1.688E-175	100.0	0	0	0.205	C:A
20	marker_20	1461490	0.004	0.387	1.7511E-182	99.9	0	0	0.273	A:G
21	marker_21	1483008	0.004	0.238	1.7511E-124	100.0	0	0	0.137	G:A
22	marker_22	1483058	0.001	0.107	1.531E-68	99.9	0	0	0.052	A:G
23	marker_23	1484632	0.002	0.189	1.6374E-106	99.6	0	0	0.126	G:A
24	marker_24	1485494	0.004	0.352	6.0687E-166	98.3	0	0	0.228	G:A
25	marker_25	1870690	0.038	0.228	1.756E-84	99.3	0	0	0.131	G:A
26	marker_26	1881207	0.005	0.466	6.5164E-204	100.0	0	0	0.369	G:A
27	marker_27	1883270	0.038	0.291	3.3146E-106	98.9	0	0	0.172	A:G
28	marker_28	1986674	0.008	0.45	7.8637E-196	99.7	0	0	0.242	A:G
29	marker_29	2114294	0.008	0.488	3.7069E-200	99.9	0	0	0.474	A:G
30	marker_30	2409577	0.008	0.485	4.2863E-209	100.0	0	0	0.444	G:A
31	marker_31	2766471	0.005	0.293	1.3569E-175	100.0	0	0	0.158	A:G
32	marker_32	2865776	0.001	0.095	2.4568E-75	99.9	0	0	0.054	A:G
33	marker_33	2998277	0.004	0.337	1.5606E-161	100.0	0	0	0.215	G:A



Number of generations: 5 | No mutations

#	Name	Position	ObsHET	PredHET	HWpval	%Geno	FamTrio	MendErr	MAF	Alleles
1	marker_1	30789	0.002	0.422	1.8628E-198	99.2	0	0	0.324	G:C
2	marker_2	30838	0.002	0.42	2.4857E-188	97.2	0	0	0.222	G:A
3	marker_3	31461	0.001	0.233	1.0622E-122	99.6	0	0	0.132	G:A
4	marker_4	32475	0.002	0.256	2.6225E-182	99.2	0	0	0.298	G:A
5	marker_5	140209	0.003	0.236	1.6869E-122	99.4	0	0	0.122	A:C
6	marker_6	515015	0.003	0.274	3.5741E-138	99.3	0	0	0.162	A:G
7	marker_7	596470	0.003	0.286	7.1181E-161	99.9	0	0	0.121	A:C
8	marker_8	608683	0.002	0.153	1.720E-81	99.4	0	0	0.112	C:A
9	marker_9	662422	0.001	0.156	2.0022E-70	99.3	0	0	0.124	G:A
10	marker_10	695932	0.003	0.463	5.0906E-200	99.9	0	0	0.344	A:G
11	marker_11	764133	0.001	0.168	1.0686E-91	99.8	0	0	0.156	G:A
12	marker_12	832195	0.004	0.257	3.8656E-130	99.3	0	0	0.152	A:G
13	marker_13	840314	0.004	0.418	3.9677E-167	100.0	0	0	0.298	A:G
14	marker_14	1047561	0.008	0.408	4.3742E-180	99.9	0	0	0.285	C:G
15	marker_15	1093747	0.002	0.228	5.2938E-123	99.8	0	0	0.131	C:A
16	marker_16	1219776	0.001	0.202	1.4421E-113	99.2	0	0	0.157	C:A
17	marker_17	1222568	0.002	0.141	3.5667E-67	99.9	0	0	0.077	G:A
18	marker_18	1255163	0.014	0.396	1.6676E-166	99.1	0	0	0.247	A:C
19	marker_19	1265191	0.003	0.326	1.688E-175	100.0	0	0	0.205	C:A
20	marker_20	1461490	0.002	0.377	1.7511E-182	99.8	0	0	0.273	A:G
21	marker_21	1483008	0.004	0.278	2.7521E-122	100.0	0	0	0.137	G:A
22	marker_22	1483058	0.001	0.102	1.5322E-74	99.9	0	0	0.052	A:G
23	marker_23	1484632	0.002	0.187	7.6774E-107	99.6	0	0	0.126	G:A
24	marker_24	1485494	0.003	0.352	5.2937E-176	98.3	0	0	0.228	G:A
25	marker_25	1870690	0.028	0.233	1.0686E-91	99.3	0	0	0.137	G:A
26	marker_26	1881207	0.005	0.466	3.3163E-204	99.6	0	0	0.319	G:A
27	marker_27	1883270	0.038	0.291	3.3146E-136	98.9	0	0	0.191	A:G
28	marker_28	1986674	0.006	0.45	5.2938E-193	99.7	0	0	0.199	A:G
29	marker_29	2114294	0.008	0.488	3.7042E-180	99.9	0	0	0.479	A:G
30	marker_30	2409577	0.006	0.352	4.2863E-209	100.0	0	0	0.447	G:A
31	marker_31	2766471	0.006	0.293	7.2863E-200	99.7	0	0	0.174	A:G
32	marker_32	2865776	0.002	0.135	2.4568E-75	100.0	0	0	0.024	A:G
33	marker_33	2998277	0.004	0.337	4.6806E-166	99.7	0	0	0.245	G:A



As we can check from the data obtained when not applying any mutations, the results remain consistent with the original, the LD display shows similar results with no variations in the relations between the markers, as well as the haplotype displays, which only present some limited variations in the numeric values, but not in the relations presented originally.

The data presented also remains consistent, which is the expected results, as this way does not modify the locus of the genes nor the SNPs and the markers, but does swap the genotypes, which make the genotype of the original providers harder to trace back.

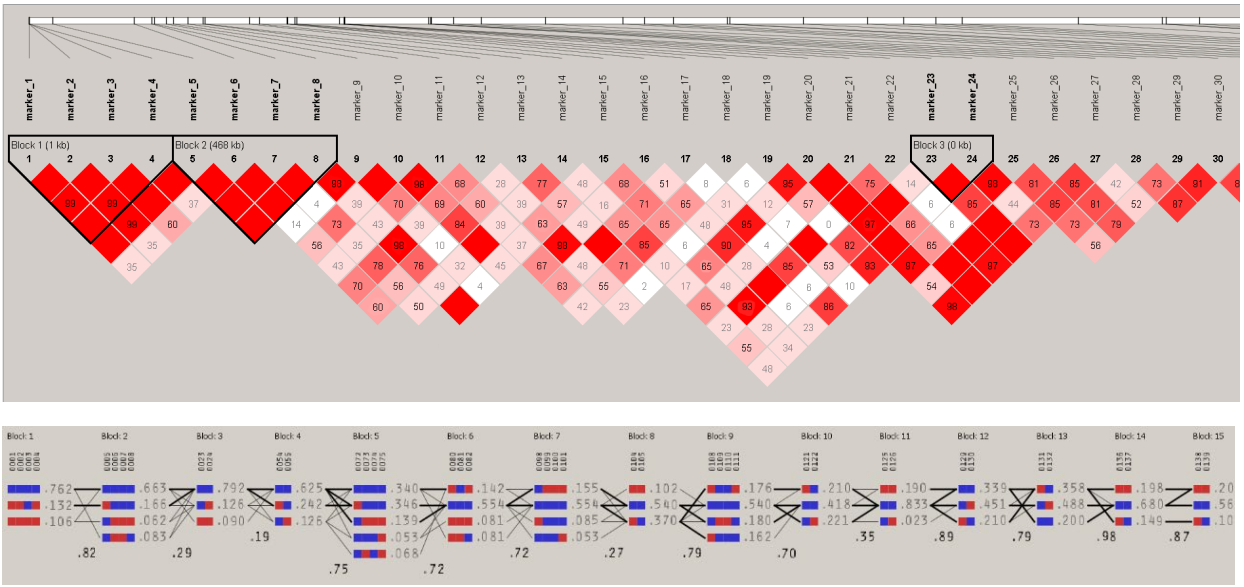
As we have seen, the number of generations does not affect the data directly, obtaining similar results, due to the markers not being modified, exchanging data in those markers of people with the same physical trait of study (in this scenario, Chron's disease).

What can make differences between generations are mutations. In nature, the transversion mutation chance to be the half value of the transition mutation chance, so we will treat it in an analogous way in our study scenarios, representing it by the variable 'mutation_chance'. It will mean the chance for a transition mutation to happen, so the transversion mutation chance would be half of that value.

Seeing how the mutations affect the sequences, the higher the mutation rate is, the more unidentifiable the data becomes, but the further the data strays from its original purpose.

We won't be looking at the Marker information segment, as we are more interested in comparing the LD display and Haplotype display graphics, as they indicate in a more visual and easier way to see how different both studied files are.

Number of generations: 5 | Mutation_Chance = $1 * 10^{-7}$



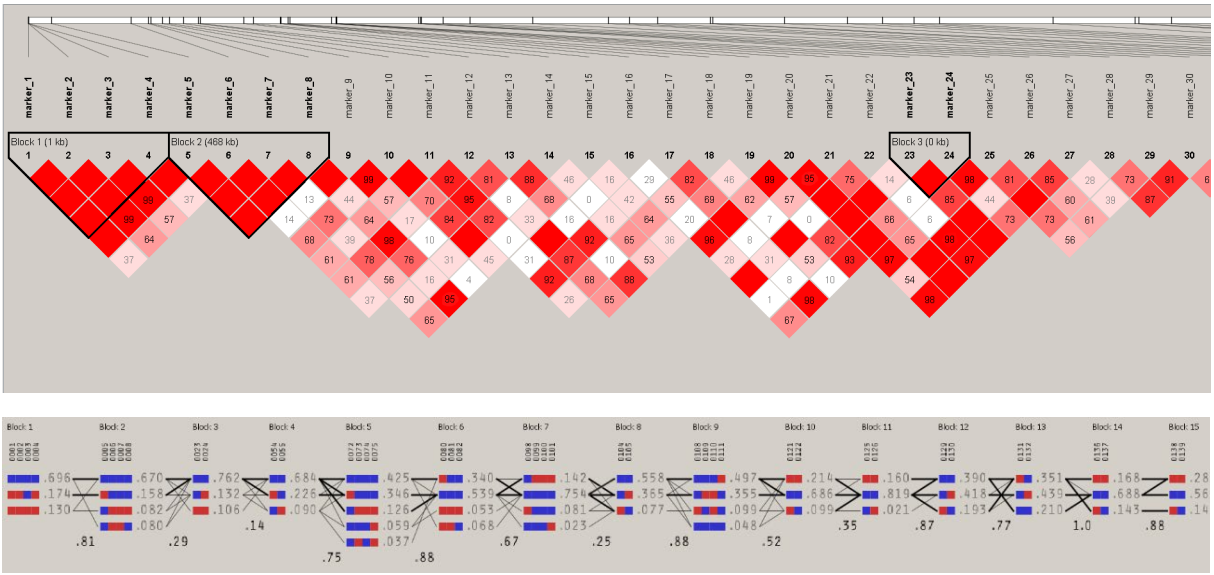
Although results are quite similar as well in this case, we can see a slight variation that happens when comparing the marker_6 with the marker_14, where a box, the Hedrick's multiallelic D value, (how we can see the way different markers associate in a non-random form), is missing, meaning that those two markers are no longer associated with each other.

Checking the block 6, we can appreciate a slight difference as well, seeing that there is now a red box in the end of the first row, whereas before we could appreciate a blue box, indicating that the allele has now been modified.

This is all due to a mutation having happened in that box, which has modified the way the markers interact, and the representation of the haplotype having been modified as a result. However, this is only a slight change, as we can check that the rest of the results retain similarity with the original ones, so this change will not manifest any problem in any future investigations, should this modified version be employed, as it is only a slight change in the whole array of data.

This is what we aim for, changing the genomic sequences in a way that makes the data unlinkable for the original donors, while still retaining the utility it can provide to investigators.

Number of generations: 5 | Mutation_Chance = $1 * 10^{-5}$

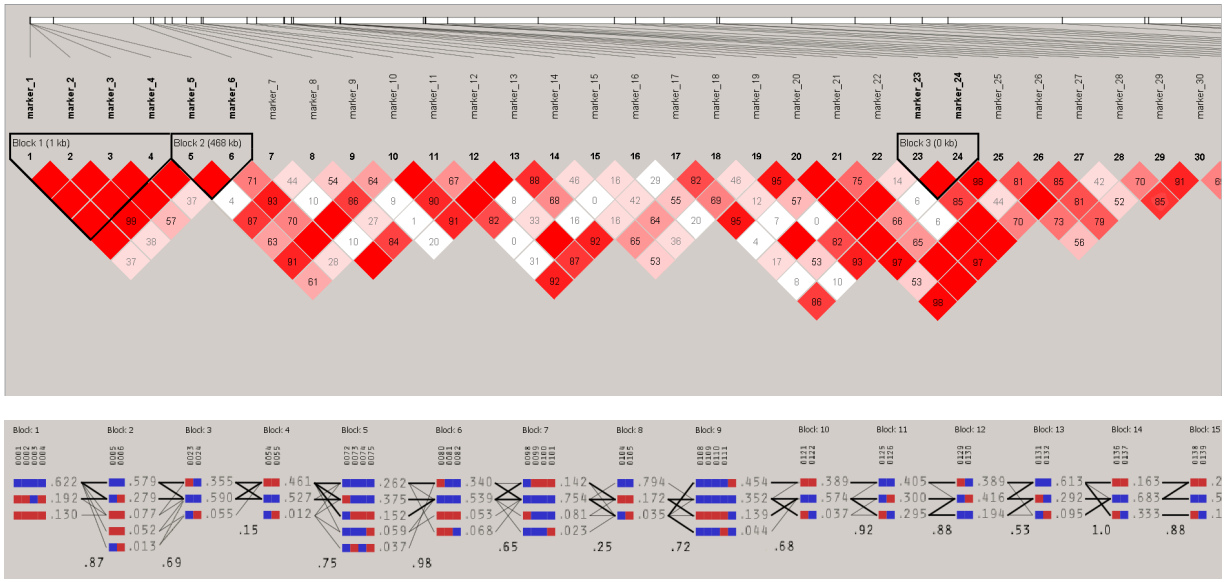


As we can see, now that the mutation rate has been increased, the changes in the LD display and the haplotype display are beginning to be increased, not only in value form, but also now modifying how the markers interact with each other, adding new interactions and removing others. For example, the LD display now shows two markers that originally interacted are no longer connected, the marker 14 and 15 with the marker 20, as the squares that indicated so are no longer present. In the haplotype display, the block 9 has received substantial changes as well, as its structure has been completely modified, as well as their interactions with its colliding blocks.

Whether this changes could add significant error or not to a study's result cannot be quantified just looking at those displays, further examination using different biotechnological methods should be employ to fully understand the significance of the alterations due to the mutations, such as employing techniques as Random Amplified Polymorphic DNA (RAPD) or Amplified Fragment Length Polymorphism (AFLP), PCR-based tools that employ DNA-fingerprinting techniques with restriction enzymes to detect polymorphisms, and thus, know if the modification has produced a new phenotype to surge.

Some commonly used software that allow to analyze AFLP data are BioNumerics [1] or SoftGenetics [2], two advanced biological data analyzing software.

Number of generations: 5 | Mutation_Chance = $1 * 10^{-4}$

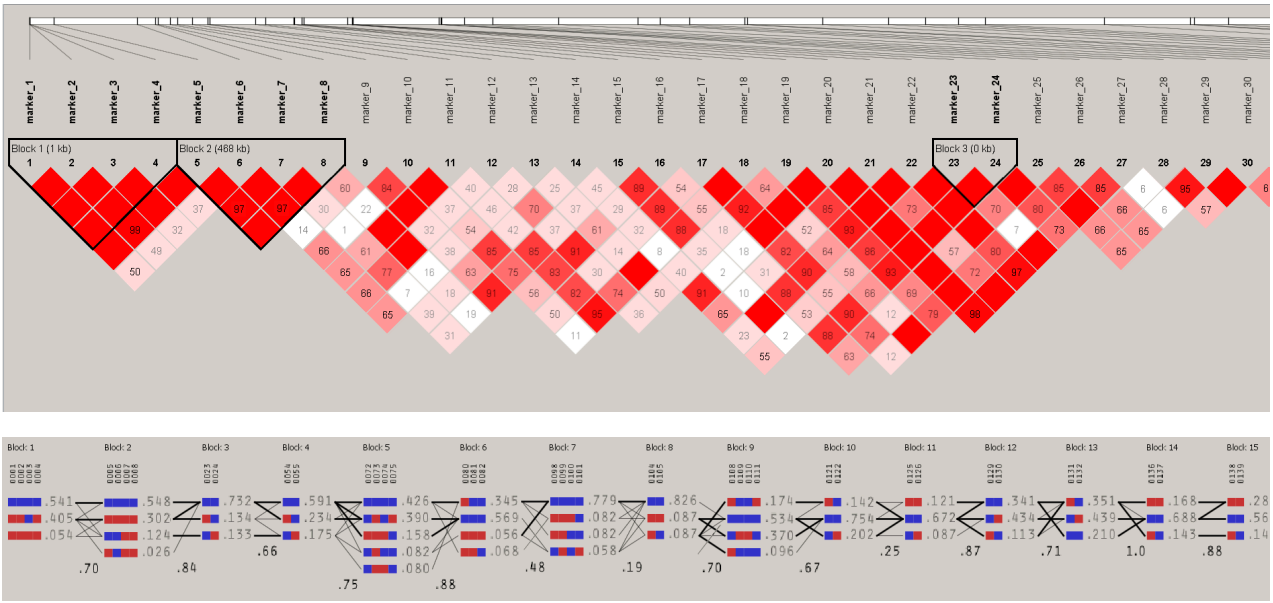


The displayed data show that now the mutations have changed greatly the different blocks and markers, displaying completely new relations, or having removed previous interactions that appeared before. The mutations have affected the markers from the marker_9 to the marker_20, showing that they are more frequent now, changing several alleles to now be associated random forms, other than having non-random relationships, shown by the 'squares' in the LD display.

This is also shown in the haplotype display, the interactions of the blocks have changed, while adding extra bases to the structure of different blocks or adding new alleles as possible representation for the block, due to the high modification that it has received, and even changing the amount of markers that are in a same block, such as it can be seen in block 2.

From this point forward, any modification done with more mutation chance provides unsuccessful results, as the chains are too modified as to the program to parse, throwing file errors due to marker errors, or having too many erroneous pairwise comparisons by the mutations.

Number of generations: 10 | Mutation_Chance = $1 * 10^{-7}$

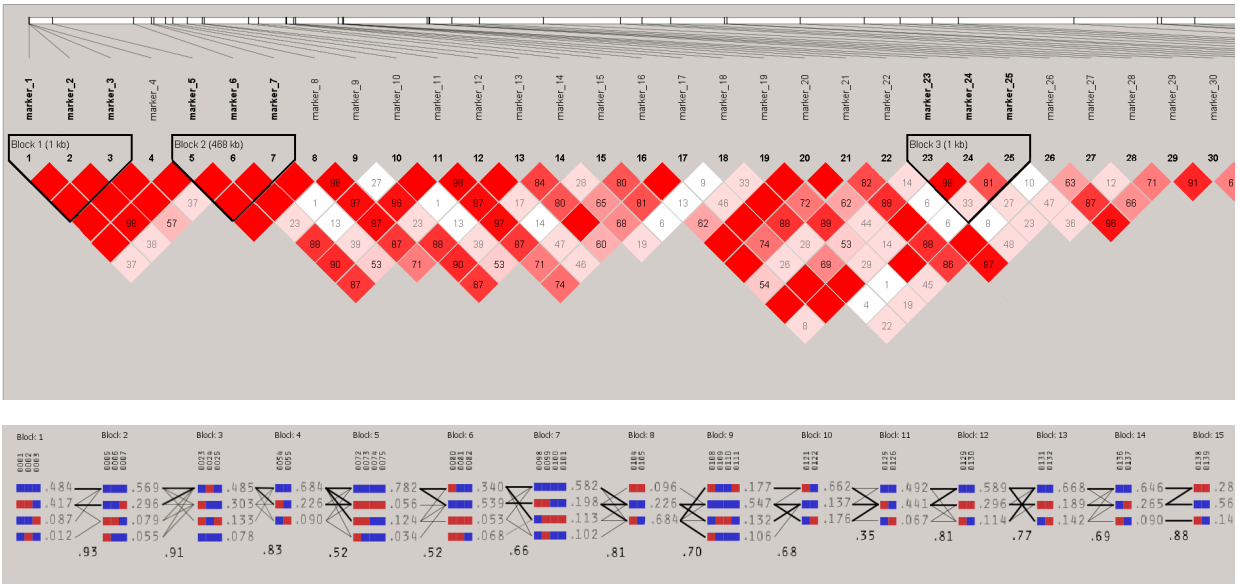


As we can see, the number of generations directly influence the quantity of mutations that happen in the genotype, having more possible rounds for mutations to appear, which can be directly represented in changes in the haplotype.

As can be seen in the haplotype display, both block 2 and block 5 present mutations that change the structures of the alleles, and the LD display shows new interactions between the markers, and others that had previously existed now aren't displayed.

As happened before, the data has changed but not in a high degree that would make it unrecognizable. However, we cannot extract how useful this data would be at first glance, the RAPD or AFLP technique should be applied as to determine this grade of mutations results in data that does not change the results of any investigations on the haplotype or not.

Number of generations: 10 | Mutation_Chance = $1 * 10^{-5}$



In this scenario, the mutations have changed greatly the structure of the haplotype due to the modifications in the genotype, the blocks changing in size now, having less markers (in the case of the first block), or more (in the case of the second block). Adding more generations with more mutation rate with more generations adds more mutations, risking changing the genotype to a degree that the markers and blocks of the haplotype are modified and thus, makes the data not useful anymore for the story, as is this case's problem.

Thus, when studying how the mutations affect the haplotype with the techniques previously mentioned, we would not recommend using this configuration and any other's that would cause the increment of mutations in the genotype.

Chapter 5

5 Conclusions and future work

Managing genomic data in a secure way is a novel practice that is not well developed, as currently employed techniques require the owner to be heavily involved or to outsource to third parties.

We have studied several cryptographic protocols and schemes that are used currently in the scientific scheme to construct privacy-preserving solutions to the problems that surge when the DNA sequence data of an individual is leaked. These methods employ homomorphic encryption, CKKS, ResNet, oblivious transfer, garbled circuits, and other techniques.

We have described a method for assisting investigations in genomic data, managing the computation over encrypted data stored in the cloud while leaving the decision on admissible computations to the data owner, while anonymizing the records before being sent from the sequencing centers. This allows the owners to remain in full control of the data, while maintaining computational costs to a minimum.

This method is based on the METIS scheme by *Battke, Florian & Sak, Dilara. (2019)* while introducing improvements on the scheme to solve its previous problems. These solutions consist of the development of a synthetic microdata generation algorithm, employing a modern OT scheme with minimal communicational costs and introducing the possibility to execute secure count queries on the encrypted genomic data to facilitate SNP studies.

We have also implemented in an algorithm to anonymize the donors' DNA by constructing new genetic data sets following the principles of population genetics by mixing and hashing initial genetic material over many generations in order to create a genetically disperse future population that shares the same genetic traits in general as the initial population but holds no direct correspondence to the original individuals. The genotype is modified, but the haplotype of the population remains unmodified, thus allowing haplotype studies to not be affected.

5.1 Future work

Our future work in this project is related to implementing the METIS variation model, as well employing biotechnological methods to verify the best parameters for our synthetic microdata generation algorithm.

First, we intend to analyze the haplotype differences that surge in our results with higher mutation rates using different biotechnological methods should be employ to fully understand the significance of the alterations due to the mutations, such as employing techniques as Random Amplified Polymorphic DNA (RAPD) or Amplified Fragment Length Polymorphism (AFLP), PCR-based tools that employ DNA-fingerprinting techniques with restriction enzymes to detect polymorphisms, and thus, know if the modification has produced a new phenotype to surge.

Finally, we will implement a functional prototype of our model with a simple user interface to test its efficiency in real-live scenarios, employing data from the eICU database and making real employed queries with the database to analyze the performance of our protocol, and were it to prove successful, then we will design a more user friendly interface for the system, prepare the implementation with cloud servers, and present it to biotechnological industries and hospitals that can be interested in the system in protecting the user's DNA sequences while not hampering their investigations.

6 Bibliography

[1] "BIONUMERICS: one universal platform to store and analyze biological data | BIONUMERICS." [Online]. Available: https://www.bionumerics.com/bionumerics. [Accessed: Jan. 04, 2023]

[2] "Residual Neural Network (ResNet)." [Online]. Available: https://iq.opengenus.org/residual-neural-networks/. [Accessed: Jan. 04, 2023]

[3] "SoftGenetics." [Online]. Available: https://softgenetics.com/. [Accessed: Jan. 04, 2023]

[4] F. Guo, Y. Mu, and W. Susilo, "Subset membership encryption and its applications to oblivious transfer," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 7, pp. 1098–1107, 2014, doi: 10.1109/TIFS.2014.2322257.

[5] A.-J. Lai, C.-K. Chu and W.-G. Tzeng, "Efficient k-out-of-n Oblivious Transfer Schemes."

[6] "An example of the LINKAGE format A toy example of a pedigree file in... | Download Scientific Diagram." [Online]. Available: https://www.researchgate.net/figure/An-example-of-the-LINKAGE-format-A-toy-example-of-a-pedigree-file-in-the-LINKAGE-format_fig1_224970383. [Accessed: Jan. 04, 2023]

[7] "Home - dbGaP - NCBI." [Online]. Available: https://www.ncbi.nlm.nih.gov/gap/. [Accessed: Nov. 22, 2022]

[8] "Haplotype mapping (Genomics)." [Online]. Available: https://what-when-how.com/genomics/haplotype-mapping-genomics/. [Accessed: Jan. 04, 2023]

[9] "Browsing Linkage Disequilibrium | Broad Institute." [Online]. Available: https://www.broadinstitute.org/haploview/browsing-linkage-disequilibrium. [Accessed: Jan. 04, 2023]

[10] "An example of the LINKAGE format A toy example of a pedigree file in... | Download Scientific Diagram." [Online]. Available: https://www.researchgate.net/figure/An-example-of-the-LINKAGE-format-A-toy-example-of-a-pedigree-file-in-the-LINKAGE-format_fig1_224970383. [Accessed: Nov. 22, 2022]

[11] "National Center for Biotechnology Information." [Online]. Available: https://www.ncbi.nlm.nih.gov/. [Accessed: Nov. 22, 2022]

[12] "Haplotype mimicking (Genomics)." [Online]. Available: https://what-when-how.com/genomics/haplotype-mimicking-genomics/. [Accessed: Nov. 22, 2022]

[13] "Browsing Linkage Disequilibrium | Broad Institute." [Online]. Available: https://www.broadinstitute.org/haploview/browsing-linkage-disequilibrium. [Accessed: Nov. 22, 2022]

[14] J. C. Barrett, B. Fry, J. Maller, and M. J. Daly, "Haploview: analysis and visualization of LD and haplotype maps," *Bioinformatics*, vol. 21, no. 2, pp. 263–265, Jan. 2005, doi: 10.1093/BIOINFORMATICS/BTH457. [Online]. Available:

https://academic.oup.com/bioinformatics/article/21/2/263/186662. [Accessed: Nov. 22, 2022]

[15] M. Slatkin, "Linkage disequilibrium — understanding the evolutionary past and mapping the medical future," *Nature Reviews Genetics 2008 9:6*, vol. 9, no. 6, pp. 477–485, Jun. 2008, doi: 10.1038/nrg2361. [Online]. Available: https://www.nature.com/articles/nrg2361. [Accessed: Nov. 22, 2022]

[16] "8.10: Sinapsis e Invasión de Hilos Individuales - LibreTexts Español." [Online]. Available: https://espanol.libretexts.org/Biolog%C3%ADa/Gen%C3%A9tica/Libro%3A_Trabajar_con_Gen%C3%A9tica_Molecular_(Hardison)/Unidad_II%3A_Replicaci%C3%B3n%2C_Mantenimiento_y_Alteraci%C3%B3n_del_Material_Gen%C3%A9tico/8%3A_Recombinaci%C3%B3n_de_ADN/8.10%3A_Sinapsis_e_Invasi%C3%B3n_de_Hilos_Individuales. [Accessed: Nov. 22, 2022]

[17] S. M. Carr, *Transition versus Transversion mutations*. Memorial University of Newfoundland [Online]. Available: https://www.mun.ca/biology/scarr/Transitions_vs_Transversions.html. [Accessed: Nov. 22, 2022]

[18] "Haplotype extraction and LD visualization in Haploview." [Online]. Available: https://avikarn.com/2021-05-18-haploview1/. [Accessed: Nov. 22, 2022]

[19] "Haploview | Broad Institute." [Online]. Available: https://www.broadinstitute.org/haploview/haploview. [Accessed: Nov. 22, 2022]

[20] J. Mu, Y. Guo, F. Chen, and R. Ma, "Efficient k-out-of-n oblivious transfer scheme with the ideal communication cost," *Theor Comput Sci*, vol. 714, pp. 15–26, 2018 [Online]. Available: https://ro.uow.edu.au/eispapers1https://ro.uow.edu.au/eispapers1/966. [Accessed: Nov. 22, 2022]

[21] "Quality (Phred) scores." [Online]. Available: https://www.drive5.com/usearch/manual/quality_score.html. [Accessed: Nov. 22, 2022]

[22] "7.1 FASTA and FASTQ formats | Computational Genomics with R." [Online]. Available: https://compgenomr.github.io/book/fasta-and-fastq-formats.html. [Accessed: Nov. 22, 2022]

[23] "Is privacy the price of precision medicine? | OUPblog." [Online]. Available: https://blog.oup.com/2017/03/privacy-precision-medicine/. [Accessed: Nov. 22, 2022]

[24] S. Suchkov *et al.*, "Personalized and precision medicine (PPM) as a model of healthcare of the newest generation: The motivation for biopharma and translational resources to move ahead together," *International Journal of Drug Development and Research*, vol. 10, 2018, doi: 10.21767/0975-9344-C1-001. [Online]. Available: https://www.itmedicalteam.pl/proceedings/personalized--precision-medicine-ppm-as-a-model-of-healthcare-of-the-newest-generation-the-motivation-for-biopharma-and--62923.html. [Accessed: Nov. 22, 2022]

[25] M. Kantarcioglu, W. Jiang, Y. Liu, and B. Malin, "A cryptographic approach to securely share and query genomic sequences," *IEEE Transactions on Information Technology in Biomedicine*, vol. 12, no. 5, pp. 606–617, 2008, doi: 10.1109/TITB.2007.908465.

[26] "What Is Homomorphic Encryption?" [Online]. Available: https://www.techtarget.com/searchsecurity/definition/homomorphic-encryption. [Accessed: Nov. 22, 2022]

[27] D. Wu, "Somewhat Homomorphic Encryption Practical."

[28] "Yao's Garbled Circuit," *CS598* [Online]. Available: https://courses.engr.illinois.edu/cs598man/fa2009/slides/ac-f09-lect16-yao.pdf. [Accessed: Nov. 22, 2022]

[29] "Garbled circuit - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Garbled_circuit. [Accessed: Nov. 22, 2022]

[30] "What Is a Message Authentication Code (MAC)? | Fortinet." [Online]. Available: https://www.fortinet.com/resources/cyberglossary/message-authentication-code. [Accessed: Nov. 22, 2022]

[31] "Oblivious transfer - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Oblivious_transfer#1-out-of-n_oblivious_transfer_and_k-out-of-n_oblivious_transfer. [Accessed: Nov. 22, 2022]

[32] B. Kaliski, "Pseudorandom Function," *Encyclopedia of Cryptography and Security*, pp. 485–485, Sep. 2005, doi: 10.1007/0-387-23483-7_329. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007/0-387-23483-7_329. [Accessed: Nov. 22, 2022]

[33] J. W. Lee *et al.*, "Privacy-Preserving Machine Learning with Fully Homomorphic Encryption for Deep Neural Network," *IEEE Access*, vol. 10, pp. 30039–30054, Jun. 2021, doi: 10.48550/arxiv.2106.07229. [Online]. Available: https://arxiv.org/abs/2106.07229v1. [Accessed: Nov. 22, 2022]

[34] "Personalized Medicine." [Online]. Available: https://www.genome.gov/genetics-glossary/Personalized-Medicine. [Accessed: Nov. 22, 2022]

[35] "Anonymizing patient genomic data for public sharing association studies - PubMed." [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/23920753/. [Accessed: Nov. 22, 2022]

[36] "Incentivizing data donation," *Nat Biotechnol*, vol. 33, no. 9, p. 885, Sep. 2015, doi: 10.1038/nbt.3341.

[37] R. Agrawal and C. Johnson, "Securing electronic health records without impeding the flow of information," *Int J Med Inform*, vol. 76, no. 5–6, pp. 471–479, May 2007, doi: 10.1016/j.ijmedinf.2006.09.015.

[38] "2.1 Steps of (genomic) data analysis | Computational Genomics with R." [Online]. Available: https://compgenomr.github.io/book/steps-of-genomic-data-analysis.html. [Accessed: Nov. 22, 2022]

[39] J. Batley and D. Edwards, “Genome sequence data: Management, storage, and visualization,” *Biotechniques*, vol. 46, no. 5 SPEC. ISSUE, pp. 333–335, Apr. 2009, doi: 10.2144/000113134/ASSET/IMAGES/LARGE/FIGURE1.JPEG. [Online]. Available: https://www.future-science.com/doi/10.2144/000113134. [Accessed: Nov. 22, 2022]

[40] M. M. al Aziz *et al.*, “Privacy-preserving techniques of genomic data—a survey,” *Brief Bioinform*, vol. 20, no. 3, pp. 887–895, May 2019, doi: 10.1093/BIB/BBX139. [Online]. Available: https://academic.oup.com/bib/article/20/3/887/4600334. [Accessed: Nov. 22, 2022]

[41] Deuber, Dominic, Christoph Egger, Katharina Fech, Giulio Malavolta, Dominique Schröder, Sri Aravinda Krishnan Thyagarajan, Florian Battke and Claudia Durand. “My Genome Belongs to Me: Controlling Third Party Computation on Genomic Data.” Proceedings on Privacy Enhancing Technologies 2019 (2018): 108 - 132. [Online]. Available: https://academic.oup.com/bib/article/20/3/887/4600334. [Accessed: Nov. 22, 2022]