Dominic Deuber, 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

Abstract: 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. Currently known solutions for privacy-respecting computation require the owner of the DNA to either be heavily involved in the execution of a cryptographic protocol or to completely outsource the access control to a third party. This motivates the demand for cryptographic protocols which enable computation over encrypted genomic data while keeping the owner of the genome in full control. We envision a scenario where data owners can exercise arbitrary and dynamic access policies, depending on the intended use of the analysis results and on the credentials of who is conducting the analysis. At the same time, data owners are not required to maintain a local copy of their entire genetic data and do not need to exhaust their computational resources in an expensive cryptographic protocol.

In this work, we present METIS, a system that assists the computation over encrypted data stored in the cloud while leaving the decision on admissible computations to the data owner. It is based on garbled circuits and supports any polynomially-computable function. A critical feature of our system is that the data owner is free from computational overload and her communication complexity is independent of the size of the input data and only linear in the size of the circuit's output. We demonstrate the practicality of our approach with an implementation and an evaluation of several functions over real datasets.

Keywords: Secure Multi-Party Computation, Protocols, DNA security, Genome Privacy

DOI 10.2478/popets-2019-0007
Received 2018-05-31; revised 2018-09-15; accepted 2018-09-16.

Dominic Deuber, Christoph Egger, Katharina Fech, Giulio Malavolta, Dominique Schröder, Sri Aravinda Krishnan Thyagarajan: Friedrich-Alexander-Universität Erlangen-Nürnberg
Florian Battke, Claudia Durand: CeGaT GmbH

1 Introduction

The DNA sequence of each individual is an unparalleled piece of information: Not only does it uniquely identify every individual, it also contains the blueprint of that person and allows predictions about physical features, health risks and much more. Working with genetic information promises groundbreaking medical advances and at the same time attracts devastating misuses.

Medical practitioners and researchers set high hopes on the analysis of genomic data: Screening the DNA sequence for specific biomarkers allows early detection and diagnosis of hereditary diseases [40]. There are also many well-known genetic risk factors that can be detected long before the disease onset, with the benefit of allowing positive lifestyle changes as well as early participation in preventive examination to completely prevent the disease through timely treatment. For example, BRCA1 and BRCA2 are human tumour suppressor genes, mutations of which strongly increase the risk of developing tumours. Medical studies have shown that women who carry mutations in one or both of these genes can have a risk of up to 80% of being diagnosed with breast cancer during their lifetimes [1]. Additionally, research in the field of personalized medicine [63] promises considerable more effective treatment.

As the price of DNA sequencing has dramatically dropped in the past years, the interest in genomic data has seen a constant growth also outside the scientific community. For instance, in 2001, sequencing a person's DNA cost roughly 100,000 $ whereas today the price has dropped to well below 1,000 $ [67]. Given the developments, the current trend and the incredible value of these data, it is conjectured that a significant proportion of the population will be sequenced within the next few decades.

While the analysis of large amounts of genomic data can bring huge benefits for society, it simultaneously raises several privacy concerns, such as data ownership, data privacy, and legitimate uses of such data. For example, health insurance companies could use knowledge of genomic information of a person to estimate the expected revenue and cost of insuring this person. As genomic information is heritable, such knowledge extends to direct offsprings of that person as well as to close relatives namely the person's siblings, parents, as well as (with lower predictive power) to more distant relatives. While some of these ethical and economic concerns are already being addressed by legislative action, we see an urgent need to develop cryptographic solutions that ensure legitimate interests of both sides (data owner and data user) are met and regulatory demands can be enforced.

This means that we need to support the analysis of genomic data while simultaneously placing complete control over who has access to that data and to what end it can be analysed into the data owners' hands. To ensure data privacy, the client that wishes to analyse a genome sequence must never get direct access to the original sequence, but may only compute functions on the encrypted genome.

In this work, we present METIS, a cryptographic system that allows secure computation on encrypted genomic data with the important feature that the data owner controls the type of functions that can be computed. We formally characterize the privacy guarantees of our system and provide a comprehensive performance evaluation to confirm the practicality of our construction.

Computation on Genomic Data. The major challenge in secure computation on genomic data is performance, since human genomic data is large: The raw genomic sequence of a single individual can be encoded in 1.5 GB by representing each of the four symbols (nucleotides) of the genetic alphabet as two-bit values and taking into account that one genome copy (23 chromosomes) is roughly 3 billion symbols long and that each individual inherits two such copies from their two parents.

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 [23]. The size of a FASTQ file is in the order of tens of gigabytes.

Given a known reference sequence, it is in principle sufficient to only encode the difference between an individual's sequence and the reference, i.e., to only store the "variants" found in the individual. These are stored in VCF files [26], which in general amount to tens of megabytes in size. While this reduces the storage size quite considerably, we cannot make use of VCF files in METIS as it would leak important information: In contrast to FASTQ files, VCF file size directly corresponds to the number of variants found in the individual. Thus, we introduce a representation of genetic data that does not reveal any measure of difference to the reference genome and is tunable to the amount of information that needs to be encoded.

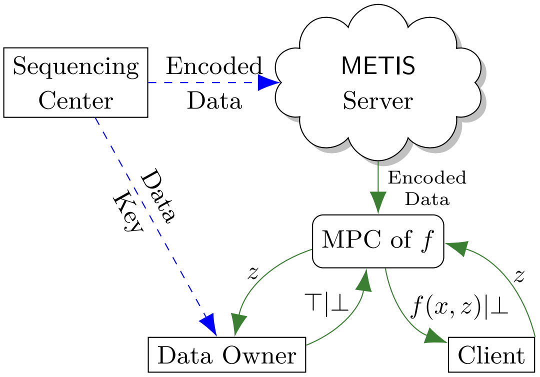

Fig. 1. High-level overview of METIS system

A High-Level Overview. The objective of METIS is to allow clients, such as medical researchers, to evaluate functions over genomic data in a controlled manner. The seemingly contradictory situation that our system resolves is as follows: We would like to keep the owners of the genome in full control over the information that is disclosed, while at the same time not straining them with the computational burden of expensive cryptographic protocols. Ideally, the owners' communication and computation complexity should be independent from the size of the genomic information.

We give an overview of the components and participants in METIS with the help of Figure 1. The genomic information is collected at the sequencing center, which encodes and stores them on the server. Note that the information stored on the server looks random to the eyes of those who do not know the corresponding decoding key. The sequencing center sends the (small) decoding key to the data owner, and erases all the local data. At any time clients can engage with the server and the data owner in a multi-party computation protocol to learn the output of a certain function f over the genomic data x and the client's query z. We stress that, while the decision of whether to allow the computation of the function on a particular query depends exclusively on the data owner, the computationally intense tasks are performed by the server and the client. At the end of the protocol the client learns the output of f on query z (if allowed by the owner) and nothing else, whereas the server learns nothing.

1.1 METIS in the real world

Given both the incredible potential of genetic research and the worrisome privacy implications, it is important for a proposed system to not only provide a cryptographically sound solution but also to balance real world interests. METIS fits well within the framework of participant-centered initiatives (PCI) [46] and the feasibility of the METIS system is evaluated in a field study. PCIs are desirable both for the participants and for researchers alike and can be considered the future of genetic research [46].

In a recent study [45], 48 % of the respondents were willing to provide their genetic information unconditionally to any study approved by an institutional ethics committee while 42 % wanted to be asked for every individual study where their genetic information was supposed to be used. Only 10 % want to specify a policy – the approach most commonly found in cryptographic protocols like Controlled Functional Encryption [62] addressing genetic privacy. Additionally, Kaufman et al. [45] identified several relevant dimensions such a policy would need to address: Apart from the privacy risks by participating, respondents also deeply cared about who is doing the research, favoring academic research with 90 % of participation over industry research where only 75 % would still want to participate, as well as the goal of the research. A one-time consent would need to include all of these constraints simultaneously.

From the perspective of researchers, a participant-centered cryptographic protocol is desirable as well. Both participation rate and privacy concerns are heavily biased [45, 46]. There are social groups who are significantly more concerned about medical privacy and reluctant to participate. Involving people in a case-by-case basis though makes them feel more involved and respected [45, 46] and sets up a conversation context between researchers and participants. Quality genetic research increasingly needs such a context between researchers and selected participants [46] to provide additional information and medical records as well as convincing candidates to participate in clinical studies.

The METIS system can be adopted to offer such a context with candidate participants. After the participant has agreed to have her genetic data evaluated for the study she can be contacted (via a pseudonym controlled by METIS) and asked to volunteer to reveal her identity and participate in follow-up research. In such a system, the server is still oblivious to the actual genetic information and the client only learns as much as the data owner agrees to reveal.

METIS requires participants to regularly interact with the system to agree to new studies. In our vision for the METIS system, data owners usually can be reached within a week to provide their consent. Especially if follow-up interaction is needed in a study these delays are unavoidable. While this is slightly inconvenient for researchers, the success of 23andMe [3] using Apple's ResearchKit [4] indicates this is indeed feasible. Typical sample sizes for genetic studies range from less than 1,000 up to 20,000 samples [37]. This is well within the range of what METIS can process in reasonable time.

1.2 Our Contribution

We propose METIS, a fully generic construction for genomic multi-party computation which is based on garbled circuits. We demonstrate the feasibility of our approach with an efficient implementation. The fundamental cryptographic building blocks of our scheme are instantiated in the standard model and therefore we do not rely on any heuristics, such as the random oracle model [16].

Using METIS, we give the data owner complete control over the genomic computation while maintaining minimal – in particular independent of the size of the genome – communication and computation. The data owner is only required to give permission to the client to receive the output of a query. An experimental evaluation shows that our system allows one to compute appropriate functions over genomic data while providing strong security guarantees. As an example, our system can securely compute whether a variant of a gene is expressed in a dataset of 10,000 participants in approximately 40 seconds.

1.3 Related Work

We review the recent related works to this paper.

Secure Function Evaluation. Kamara et al. [41, 42] provide protocols for single-server-aided Secure Function Evaluation (SFE). By outsourcing the evaluation of the garbled circuit to the server the evaluator's computational amount is reduced to be linear in its input and not in the circuit. Beforehand the parties have to run a coin-tossing protocol to share a secret key. Carter et al. [20] propose a model for SFE in a mobile device based environment. Mood et al. [58] propose the use of partial computation in the same setting. Both consider a computation among two users with one of them as a garbled circuit generator and the mobile device as the circuit evaluator. To facilitate ease of evaluation given the low computational resource of a mobile device, the computation is outsourced to a cloud which is considered a non-colluding entity. The parties know each other ahead of time as the mobile device is required to be online and to participate in an oblivious transfer with the client. This contrasts to our setting where the cloud is the circuit generator and the data owner (the mobile device) is only involved in lending permission to any client request. Moreover, there is no preprocessing involving the data owner.

Jakobsen et al. [36] proposed a framework for outsourcing secure computation where a single client with inputs outsources computation in a secure manner to possibly untrusted workers. Their framework considers a security model that has all but one worker to be malicious. This is in contrast with our model where the data owner is not supposed to learn the output of the computation.

Carter et al. [19] presented Whitewash, a protocol to outsource garbled circuit generation in mobile device constrained environments. In their setting however, the mobile device is the one generating garbled inputs and the randomness to the circuit for every computation.

Secure Computation over Genomic Data. Advancements in secure function evaluation have recently produced several efficient protocols for the evaluation of specific functionalities over genomic data, such as edit distance comparison, private-set intersection, and Hamming distance [11, 28, 38, 47, 65, 66]. Although they achieve surprising results, all of these works address specific problems and do not quite satisfy our need for a fully generic system where clients are allowed to compute *any* polynomially-computable function over genomic data.

One possible way to solve the problem of the data owner having to store her genomic data locally is to outsource this data and the burden of the computation to a semi-trusted party. This solution has been extensively studied in literature [8–10, 43], and the main idea behind these approaches is to store data on the cloud in an encrypted form. Any client can request the server for some function that the server can compute leveraging homomorphic properties of the encryption scheme. Unfortunately, these settings have the shortcoming that they only consider affine functionalities to be evaluated over the homomorphic encryption. For general functionalities they have to either use fully homomorphic encryption which is highly inefficient for practical purposes or add interactions which is not desirable.

Naveed et al. [62] proposed a new primitive called Controlled Functional Encryption (CFE) where computations can be performed on encrypted data in a controlled manner, i.e. only if the function satisfies the policy of the data owner. In their setting the data owner goes offline after outsourcing her dataset and delegates some semi-trusted authority (analogous to METIS server in our setting) for the enforcement of her policy. As discussed before, one-time policy commitments may not be the most desirable solution for real world applications as they do not match the needs of the data owners. In our setting data owners do not have to fully specify their policy but are allowed to take decisions completely freely when requests arise. This is important as the data owner's privacy preferences might change over time. In contrast to CFE, METIS can hide the policy (and all changes) even against the server. Hiding changes is essential in cases where they might be caused by events the data owner wishes to keep private. Additionally, data owners do not have to rely on any semi-trusted party to enforce their policy.

Karvelas et al. [44] proposed an ORAM based system for computations on genomic data. However, this system allows only basic access policies. The data owner grants permission to a party to perform computations on the requested data solely based on the access privileges of the party's identity. In contrast to our system, the data owner has no control over which function is computed as she does not learn the function and is not involved in the function evaluation after granting the access permission. Furthermore, the cloud service provider of their system requires two non-colluding entities acting together. In our setting there is only one entity performing the role of the service cloud provider.

Secure Data Exchange. Gilad-Bachrach et al. [30] introduced a generic system to outsource computation to a semi-trusted cloud provider that has similar characteristics to our system. The crucial difference from our proposal is that their work requires a number of oblivious transfers (OT) which is linear in the size of the inputs, whereas in our system the number of OTs is linear in the encoding of the function. Recall that OT protocols are known only from public-key primitives and are typically computationally expensive. This feature becomes critical when the size of the inputs is large (such as a genomic sequence) and the functions that we consider admit a succinct representation.

For this reason, the scheme of Gilad-Bachrach et al. heavily relies on OT extensions [35] to build an efficient system. However, efficient OT extensions are only known to exist in the random oracle model which is not sound [18]. While this is an acceptable trade-off to gain efficiency for generic outsourced computation – protocols in the random oracle model are commonly used – genetic data is especially critical information and an efficient solution that only relies on standard assumptions is desirable.

Garbled Circuits. Garbled circuits were introduced in the seminal work of Yao [68]. Lindell and Pinkas [55] provided the first proof for Yao's protocol. Later Bellare et al. [14] formalized the notion with appropriate security definitions and security proofs. Applications and practical instantiations of Yao's garbled circuits can be found in [13, 29, 32, 56]. Much work has been going on recently related to optimizations and better security for garbled circuits that can be found in [22, 33, 34, 48, 49, 51–53, 71].

2 System Overview

We envision the METIS system as a service provider between genetic researchers and the data owners. The server in METIS stores the genetic data for many data owners and provides researchers with an opportunity to evaluate diagnostic functions on a large set of genetic samples. This can be used for example to select a set of people carrying a specific genetic variant for a clinical study on a new medication. The server therefore needs to be able to evaluate the researchers' queries on thousands of stored genetic datasets. The challenge here is to avoid any leakage of information beyond what the data owner has authorized.

Settings. The METIS system consists of four types of parties with designated roles. The parties are a *sequencing center*, a *server*, *data owners* and *clients*. We simplify the following discussion by assuming the existence of a secure and authenticated channel between all parties. We also assume all parties are aware of the public keys of the other parties involved and therefore we implicitly assume them to be authenticated (e. g., via a public-key infrastructure). The same security can be easily achieved by using the server as a relay and encrypting the data with the public key of the recipient, under a non-malleable encryption scheme [25]. The METIS protocol is structured in two phases: In the *setup phase*, the sequencing center produces the DNA sequence and interacts with the server and the data owner to store the genetic information. In the *evaluation phase*, the client interacts with the server and the data owner to evaluate a function over the genetic information. We require that the computation and the communication flow of the data owner is succinct, in particular, it should be independent of the size of the genome.

In the real world METIS could be an interface operated by some national health institute to grant legitimate researchers access to a body of genetic information. The clients are researchers (from academia, government and industry) registered with the institute and data owners are volunteers. Example uses of the METIS system would be sampling potential participants for a follow-up clinical trial of new medication.

Challenges and Techniques. The first difficulty in the design of our system stems from the fact that we want to keep the data owner in control for the whole time. To facilitate the participation, we envision that the owners should be able to run the system on resource-constrained devices, such as mobile phones. Although high-end smartphones are equipped with multiple powerful cores, performing intensive computation tasks on the scale of genomic data would still drain the battery significantly. This immediately rules out the scenario where the data owner stores the genomic information locally and privately computes the queries of the clients. We also argue that this is undesirable for several reasons: First, it burdens the owner with computation over the genome each time some client issues a query. Additionally, this assumes the capability of each user to setup a reliable infrastructure to run multi-party computation protocols, without any accidental disclosure of data. It follows that existing multi-party computation solutions cannot be used off-the-shelf, as they normally share the load of computation and communication equally among all the parties.

Our construction achieves this goal by devising a custom protocol for data encoding and leveraging the security and integrity properties of Yao's garbled circuits (GC) [68]. Garbled circuits constitute an invaluable tool for secure multi-party computation: Given a suitable encoding of the inputs, they allow to securely compute any function, without revealing the inputs. Throughout the past years, garbled circuits have gone through a path of optimizations that made them efficient enough for practical applications.

Cryptographic System. In the following we provide a high-level overview of the design choices of our system. A pictorial description is given in Figure 2. In the setup phase, the sequencing center draws a secret key for a pseudorandom function and evaluates the function to generate a series of pseudorandom labels. Then it samples a sequence of random labels of the same length. The two sequences of labels constitute an encoding information: For every position i we store a pair of labels $\left(\ell_i^0, \ell_i^1\right)$. If the bit of the binary representation of the genomic sequence at position i is 0, then ℓ_i^0 is set to be a pseudorandom label whereas ℓ_i^1 is a truly random label, and vice versa if the bit is 1. 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. However, it does not, on its own, reveal any information about the original input (the corresponding original bitstring).

Each time a client wants to evaluate a function on the owner's DNA sequence, it signals the intent to the server. The server then informs the data owner about the request and generates a garbled circuit given the encoding provided by the sequencing center. The server forwards the decoding information for the circuit to the data owner and the circuit to the client. Meanwhile the owner sends the key to reconstruct the encoded input to the client, who can now evaluate the circuit. If the owner wishes to let the client evaluate the desired function (which can be parsed from the decoding table), then it sends also the decoding information to the client. Note that the output of a garbled circuit without the decoding information, consists of a set of randomly chosen bitstrings.

For efficiency reasons we split the sequenced DNA string into blocks based on biological units (like genes). Computing on one block is clearly less expensive and in most of the cases the function privacy provided by this simplified scheme is sufficient. We note, however, that this does not affect the generality of our approach, since one can represent the entire DNA sequence as one block.

Threat Model. Our corruption model follows (and strengthens) the non-collusion paradigm of [62]. In our system, the client wishes to evaluate functions on the genetic data and some input of its choice. As its input may contain sensible business logic, it must be hidden from the eyes of the METIS system. At the same time, the data owner needs to learn those inputs to make an informed decision on whether to engage in the evaluation. Recall that data owners are ordinary people who do not maintain a high-security computing device, so it is desirable that a compromise of their computer alone does not reveal anything about the genetic data. We model the client and the data owner as potentially malicious, but not colluding. This captures the scenario where a client may try to learn more information than he is supposed to and the case where the owner's machine is infected by malware.

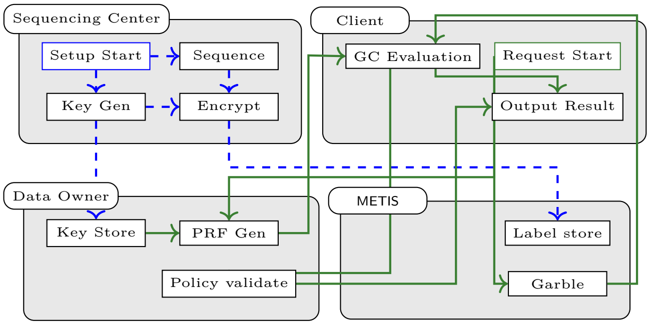

Fig. 2. Structure of the METIS system

The server acts as a mediator between the parties. It is involved in all the interactions and stores the encrypted genetic data but should not learn anything about the actual genome or the input of the client. For efficiency reasons, we model the server as a (non-colluding) semi-honest party. This is justified by the observation that the only harm that a fully corrupted server may cause is the generation of a faulty circuit which will output the wrong result to the client. We stress that even a malicious server cannot learn any information about the genomic data. All of the parties of our system are assumed to be authenticated, e. g., via a public-key infrastructure.

Performance Evaluation. We show that an evaluation of plausible diagnostic functions on the data set in our setting is possible even for a large number of data owners within a reasonable amount of time. For instance, on a 48-core server it takes approximately 40 seconds to check for a specific genomic variant at a fixed position using a dataset of 10,000 participants.

3 Preliminaries

We introduce the notation that is used throughout this work. Let λ denote the security parameter. A probabilistic polynomial time (PPT) algorithm is defined as a Turing machine that runs at most in polynomial many steps in λ for all inputs. We denote by $\mathsf{negl}(\lambda)$ a negligible function in λ if the function negl is smaller than the inverse of any polynomial for sufficiently large λ. The symbol $\approx$ stands for computational indistinguishability. When obvious, we omit the randomness from the inputs of the algorithms and therefore by $y \leftarrow \mathcal{A}(x)$ we mean that for random coins r and on input x, the output of running a PPT algorithm $\mathcal{A}$ is bound to y. By $x \leftarrow_{\$} S$ we mean that x is chosen uniformly at random from a finite set S. An algorithm having black-box access to a subroutine $\mathcal{R}$ is represented as $\mathcal{A}^{\mathcal{R}}$. We use $\mathbf{send}_P(x)$ to denote the sending of a message x to some party P and $\mathbf{receive}_P(x)$ to denote the receiving of a message x from some party P. The special symbols $\top$ and $\perp$ are used as distinguished symbols that are not strings, meaning true and false, respectively. The set of integers from 1 to n is represented by $[n]$. Finally, for two strings a and b we write $a||b$ to denote their concatenation and $a \oplus b$ to denote their bit-wise XOR.

3.1 Biological Background

Genetic information is encoded by the sequence of four different nucleotides, called adenine, cytosine, guanine and thymine, and represented by the letter A, C, G, and T. The sequence of nucleotides is determined by their binding to a sugar-phosphate backbone, yielding a very long macromolecule, called deoxyribonucleic acid (DNA). DNA molecules have a property called complementarity: Due to the fact that adenine and thymine can form hydrogen bonds with each other (same holds for cytosine and guanine), each sequence of nucleotides has a well-defined reverse-complementary sequence (e.g., ATTCG is the reverse complement of CGAAT). This means that two complementary single-strand DNA molecules form a double-strand complex, which is the well-known DNA double-helix. This feature is the basis for reproduction as it allows the creation of copies of existing single-stranded DNA as well as for the high stability of genomic information, as it enables several error-correction mechanisms (e.g., one strand is damaged but the other can serve as template for re-creating the damaged sequence). Genetic information in humans is encoded in 23 separate molecules, the chromosomes: 22 of them are present in two copies (i.e., the maternal and the paternal copy), whereas the 23rd pair consists of the so-called sex chromosomes X and Y, which determine a person's biological gender (XX being the regular female complement and XY being the regular male complement).

In general, the DNA of two individuals differ in 0.1% of the positions [39] and the most common differences are single nucleotide polymorphisms (SNPs) [24]. A SNP denotes the difference in one nucleotide at a specific position in the genome. Besides SNPs there are other variations such as small insertions or deletions of nucleotide bases, and larger chromosomal events such as inversions and translocations. Since there are only few variations between two individuals' genomes, it is common practice to denote only the differences of a newly sequenced genome compared to a reference genome. This information is typically encoded in the Variant Call Format (VCF) [26]. Note that the reference genome is based on a single (haploid) chromosome set and does represent only a commonly accepted reference. It does not represent "biological correctness", a "healthy genome", nor does it completely represent all genomic sequences (due to technical limitations and some large inter-individual differences).

The genomic sequence is, to current knowledge, made up of regions with different functions. A relatively small part (about 1% of the total sequence) consists of so-called coding sequence, commonly referred to as genes. Each gene is made up of one or more consecutive subsequences of the genomic sequence that together define the template for the production of a particular protein. Variants that occur within a coding sequence most likely have direct consequences on the shape and function of a cellular protein and defects in protein function can manifest as diseases.

3.2 Cryptographic Building Blocks

We make use of several cryptographic tools in our construction of the METIS system. Below, we provide an informal description of those primitives and we introduce their notation. For a formal definition we refer the reader to Appendix A.

Pseudorandom Functions. Pseudorandom functions (PRFs) were first introduced by Goldreich et al. [31]. A PRF is defined as a keyed function for some key k, that is computationally indistinguishable from a function f

chosen uniformly at random out of the set of all functions having the same domain and range. This captures the fact that for all inputs x, the output $\mathsf{PRF}(k, x)$ looks random to the eyes of a computationally bounded adversary. Naor and Reingold [61] proposed an efficient instantiation of PRFs, and many more can be found in the literature.

Strong One-Time MACs. Message Authentication Codes (MACs) help preventing an adversary from modifying a message sent by one party to another, without the parties detecting that a modification has been made. A sender associates a message m with a MAC tag t that was generated by computing $\mathsf{MAC}(k, m)$ with a key k. The receiver, who also knows the key, verifies whether the tag is correct on the associated message by checking if $\mathsf{MACVfy}(k, m, t) = 1$. A one-time MAC denoted by OTMAC, is a restricted version of a MAC where each key can be used at most once. The strong security of a MAC scheme implies that no adversary can forge a new valid message-tag pair. One-time MAC schemes can be instantiated from pairwise independent hash functions [21].

Oblivious Transfer. Oblivious Transfer (OT) is a cryptographic protocol $(\mathsf{sender}_{\mathsf{OT}}, \mathsf{receiver}_{\mathsf{OT}})$ where the sender has two input messages m_0 and m_1 and the receiver has an input bit b. At the end of the protocol the receiver learns m_b and nothing else. The security requirement demands that the sender should not learn the bit b and the receiver should not learn anything about m_{1-b}. Several efficient instantiations are known from standard number-theoretic assumptions [7, 59, 64].

3.3 Garbled Circuits

Here we informally describe the garbling scheme of Yao [68] and we defer a formal treatment of the primitive and its security guarantees to Appendix A. Garbled circuits allow two parties (the garbler and the evaluator) holding inputs x and y, respectively, to evaluate an arbitrary Boolean circuit of a function f without revealing any information about their inputs, beyond what is trivially leaked by the output $f(x, y)$. A garbling scheme $\mathcal{G}$ is defined with respect to a function f and is composed of an encoding (information) generation algorithm GenEn, an encoding algorithm En, a complement encoding algorithm ComEn, a circuit garbling algorithm Gb, a decoding algorithm De, and an evaluation algorithm Ev.

Fixed Input. Note that we slightly deviate from the standard notation of projective garbling schemes as introduced in [15], since we require that part of the input of the circuit is fixed throughout multiple executions of Yao's protocol. A similar setting has been considered in the work of Naveed et al. [62]. A fixed input is one that is used in multiple function evaluations. Moreover in our case, the input encodings (i. e., the labels of the garbled circuit) are also fixed and reused in the garbling of different circuits. To formalize this we introduce the algorithms GenEn and ComEn.

Input Encoding. The encoding generation algorithm GenEn for standard inputs of length n samples a vector $e = \left\{\left(\ell_i^0, \ell_i^1\right)\right\}_{i\in[n]}$ of random labels. The encoding algorithm En on input e and $x = (x_1, \ldots, x_n)$ returns the encoded input $X = \left\{\ell_i^{x_i}\right\}_{i\in[n]}$. Given a fixed input $y = (y_1, \ldots, y_n)$ and its encoded version $Y = \left\{\ell_i^{y_i}\right\}_{i\in[n]}$, the complement encoding algorithm ComEn samples a vector $\left\{\ell_i^{1-y_i}\right\}_{i\in[n]}$ uniformly at random and returns the encoding $e = \left\{\left(\ell_i^0, \ell_i^1\right)\right\}_{i\in[n]}$.

Circuit Generation. The circuit garbling algorithm Gb associates two random labels ℓ_i^0, ℓ_i^1 with each wire i of the circuit. For the input wires, it uses the encoding e as defined above. For each binary gate g of the circuit with input wires i and j and output wire w, the algorithm computes

$$\mathsf{Enc}\left(\left(\ell_i^{b_i} \| \ell_j^{b_j}\right), \ell_w^{g(b_i, b_j)}\right)$$

for all $(b_i, b_j) \in \{0, 1\}^2$, where Enc is the encryption algorithm of some symmetric encryption scheme. The resulting four ciphertexts (in random order), constitute the garbled version of g. The collection of all garbled gates forms the garbled circuit. It is computed by the garbler and sent to the evaluator, along with the decoding information d, a mapping table between the labels associated with the output wires and the output bits.

Oblivious Transfer. The evaluator must receive the labels corresponding to the input bits of each party's input. Fixed labels $\left\{\ell_i^{y_i}\right\}_{i\in[n]}$ can be sent once and for all, but for the non-fixed input x, the circuit generator has to engage the evaluator in an oblivious transfer protocol to transmit the set of labels $\left\{\ell_i^{x_i}\right\}_{i\in[n]}$.

Circuit Evaluation. Given one label for each input wire of the circuit, the evaluation algorithm Ev can compute exactly one label for each output wire of the circuit. With the set of output labels and the decoding information the decoding algorithm De computes the output $f(x, y)$.

4 METIS

In the following, we describe METIS, our protocol for secure computation on genomic data. We first introduce the formal security definitions we want to achieve followed by our construction and an intuitive security analysis. The formal proofs are deferred to Appendix B.

4.1 Definitions

We specify the security of our system following the simulation paradigm: The ideal functionality $\mathcal{F}$ is defined as a trusted third party in the ideal world [54], that interacts with the participating parties namely the server M, the client C and the owner O. The parties communicate via authenticated secure channels. Since the sequencing center is considered to be trusted, we simply omit it from our formalization and let $\mathcal{F}$ impersonate the role of it. We assume that the data is split into blocks of a certain length and let such a length be a system parameter.

Corruption Model. The adversary in this setting is required to be admissible in the sense that it can corrupt exactly one of the participating parties. If the party is maliciously corrupted, the communication to and from the party is routed through the adversary who can modify the messages arbitrarily. In contrast, if the party is corrupted in the semi-honest sense then the adversary receives the transcript of all the communication that took place to and from the corrupted party and the party's random coins. The adversary chooses one of the parties to corrupt ahead of the execution and signals its choice to the environment. More precisely, the adversary is allowed to maliciously corrupt either the client C or owner O or he can corrupt the server M in a semi-honest sense.

Execution in the Ideal Model. In the following we describe the execution of the ideal functionality. For the ease of the exposition we assume that the clients query $\mathcal{F}$ on a single block of size B, although it is easy to extend our model to the more general case where we allow queries on multiple blocks.

1. $\mathcal{F}$ samples the data x according to some distribution and fixes B (which is a system parameter) as input.
2. $\mathcal{F}$ notifies M, C and O about B.
3. Whenever C sends an evaluation request (S_{ID}, z, f, j) to $\mathcal{F}$ where S_{ID} is the session ID, z is the query of C, f is the function to be computed and j is the index of the requested block, the following steps are executed:
   (a) $\mathcal{F}$ picks a random handler h and stores (S_{ID}, z, f, h) in a table. Here the handler h acts as an identifier for the query request.
   (b) $\mathcal{F}$ notifies M about $(h, |z|, f, j)$.
   (c) $\mathcal{F}$ then sends (h, z, f, j) to O.
   (d) O responds to $\mathcal{F}$ with either $\top$ or $\bot$ (allowing or denying the evaluation request, respectively).
   (e) $\mathcal{F}$ sends $(S_{ID}, f(x, z))$ or $\bot$ to C depending on whether O answered with $\top$ or $\bot$, respectively.
   (f) An honest C outputs whatever it received from $\mathcal{F}$ in the above step.

Execution in the Real World. Next, we consider the execution in the real world where the parties interact with each other and there is no trusted party. An adversary who corrupts a party (O or C) maliciously, sends and receives messages on behalf of the party in any arbitrary way of his choice. And a semi-honest adversarial corruption of a party leads to the adversary obtaining a transcript of communication during the execution of the party. However, an honest party follows the protocol specification of the system.

Definition 4.1. *We denote the joint output of the honest and corrupt parties in the ideal world with the functionality $\mathcal{F}$ and the simulator $\mathcal{S}$ by $\{\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}}(x, z, \lambda)\}_{x,z,\lambda}$, and denote protocol π's joint output of the honest and corrupt parties in the real world as $\{\mathrm{REAL}_{\pi,\mathcal{A}}(x, z, \lambda)\}_{x,z,\lambda}$. A* METIS *system is considered* secure*, if for all functions f and every non-uniform* PPT *admissible adversary $\mathcal{A}$ in the real world, there exists a non-uniform* PPT *simulator $\mathcal{S}$ in the ideal world who corrupts the same party, such that*

$$\{\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}}(x, z, \lambda)\}_{x,z,\lambda} \approx \{\mathrm{REAL}_{\pi,\mathcal{A}}(x, z, \lambda)\}_{x,z,\lambda}.$$

4.2 METIS construction

The METIS protocol is parameterized by the blocksize B, which we assume to be known by all participants. The protocol is structured in two phases: During the setup phase the sequencing center S interacts with the data owner O and the server M. At the end of this phase, M holds an encoding of the data and O the corresponding decoding information. In the evaluation phase, the client C interacts with M and O to compute a function of his interest over O's DNA. Here the data owner's sequenced DNA is fixed across all computations.

In the following we provide the reader with an intuitive description of our system and we refer to Figure 3 for the formal specifications. Throughout the following section we assume that the parties can securely communicate with each other, whereas in practice all the communication is routed through server M. This can be easily realized using authenticated encryption.

Setup Phase. At the beginning of the setup phase, the sequencing center S samples a random key k to generate keys k_j for all j blocks of the sequenced DNA denoted by x, using a pseudorandom function. Then it creates the encoding information e_x as follows: For the i-th bit of the j-th block of x, denoted by $x_{j,i}$, it computes two labels. One represents $x_{j,i}$ and is determined by the pseudorandom function with input (k_j, i). The other label (representing the bit $1 - x_{j,i}$) is a randomly sampled string. As a result, the labels corresponding to x are fixed and can be reconstructed using only the PRF and the key k. The sequencing center S sends e_x, the sequence of zero and one labels for x which are fixed, to the server M and the key k to the data owner O. Finally, S securely deletes all data.

Evaluation Phase. The evaluation is initiated by the client C who receives f and z as inputs. The client sends f to the server. Then M and C execute a modified version of Yao's protocol: M takes the role of the garbler and C evaluates the circuit. M garbles the circuit ∇ which takes as input the data x and an input z and returns $z||f(x,z)$. Note that C does not know x and therefore it cannot run a standard oblivious transfer for the labels corresponding to x with M. Instead, it uses the PRF key (sent by O) to directly reconstruct the correct labels to evaluate the garbled circuit. Note also that M has access to the encoding e_x (the zero- and one-labels) and therefore it can garble the circuit without learning which labels correspond to the input x. Finally, instead of sending the output table to the client alongside the garbled circuit, M blinds the values corresponding to the function output $d_{f(x,z)}$ with a random string v and sends this blinded decoding information including an OTMAC to the client and the blinding factor v together with the decoding information for the client's input d_z to the owner. Intuitively, with the OTMAC we prevent selective failure attacks from a corrupted data owner as any modification to the string v is detected by the client with overwhelming probability.

In parallel to the setup of Yao's protocol, the client C interacts with the owner O and retrieves the PRF key corresponding to the block of interest j from O. As the labels corresponding to the input x were originally created using the PRF on the block key k_j and the offset i, C can now reconstruct the correct input labels. Note that since k_j is a random string, it does not leak any additional information to C.

Finally, after the client C and the server M successfully evaluated the modified version of Yao's protocol, C holds the output labels corresponding to $z||f(x,z)$. The client passes the values corresponding to the request z to the owner O, who can decode and validate it according to her policy ϕ. If she agrees with the function f and the request z, she sends the blinding factor to the client who can now verify the OTMAC and decode $f(x,z)$.

At the end of the execution the client learns the output of the function and nothing else (if allowed by the data owner), whereas O learns the request that the client computed while not learning the result of the evaluation. This is desirable to protect against potential security breaches at the data owner's end.

Note that the encoding e_x is the fixed input encoding corresponding to O's sequenced DNA. For multiple queries from clients regarding the same data owner O the server M computes different garbled circuits with the same e_x. However, M samples fresh encoding information e_z for every query z from the client C even for the same data owner O.

4.3 Security Analysis

In the following we state the main theorem of this work.

Theorem 4.2. *Let* $\mathsf{PRF}(\cdot,\cdot)$ *be a secure PRF, let* $\mathsf{OTMAC} = (\mathsf{MACGen}, \mathsf{MAC}, \mathsf{MACVfy})$ *be a strongly unforgeable one-time MAC, let* $\mathcal{G} = (\mathsf{GenEn}, \mathsf{Gb}, \mathsf{En}, \mathsf{ComEn}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ *be a projective garbling scheme and let* $(\mathsf{sender}_{\mathsf{OT}}, \mathsf{receiver}_{\mathsf{OT}})$ *be a secure 1-out-of-2 oblivious transfer protocol. Then the construction in Figure 3 securely realises the ideal functionality* $\mathcal{F}$ *(Definition 4.1).*

Proof Sketch. In order to prove that the METIS system is secure, we need to describe a simulator for every corrupted party simulating its view. Here we provide the proof sketches, and the detailed proofs can be found in Appendix B.

Corrupted Data Owner. The simulator $\mathcal{S}_{\mathsf{O}}$ receives (h, z, f, j) as input, it samples a random key k and sends it to $\mathcal{A}$. It then samples $|z|$ pairs of random labels e_z,

```
Sequencing Center $\mathsf{S}(x, B)$
$k \leftarrow_{\$} \{0,1\}^{\lambda}$
for $j = 1, \dots, \left\lceil \frac{|x|}{B} \right\rceil$ do
    $k_j \leftarrow \mathsf{PRF}(k, j)$
    for $i = 1, \dots, B$ do
        $\ell_{j,i}^{x_{j,i}} \leftarrow \mathsf{PRF}(k_j, i)$
        $\ell_{j,i}^{1-x_{j,i}} \leftarrow_{\$} \{0,1\}^{\lambda}$
$e_x := \left\{ \left( \ell_{j,i}^0, \ell_{j,i}^1 \right) \right\}_{j \in \left[ \left\lceil \frac{|x|}{B} \right\rceil \right], i \in [B]}$
return $e_x, k$
```

```
Data Owner $\mathsf{O}(k, \phi)$
receive $_{\mathsf{M}}(j, f, d_z, v)$
$k_j := \mathsf{PRF}(k, j)$
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 $i = 1, \dots, |z|$ do
    $\ell_i^0 \leftarrow_{\$} \{0,1\}^{\lambda}$
    $\ell_i^1 \leftarrow_{\$} \{0,1\}^{\lambda}$
    $\mathsf{sender}_{\mathsf{OT}}\left(\left(\ell_i^0, \ell_i^1\right)\right)$
$e_z := \left\{ \left( \ell_i^0, \ell_i^1 \right) \right\}_{i \in [|z|]}$
$e := e_z || e_x$
$\nabla(x, z) := z || f(x, z)$
$(F, d) \leftarrow \mathsf{Gb}\left(1^{\lambda}, \nabla(\cdot, \cdot), e\right)$
$d_z || d_{f(x,z)} := d$
$\mathsf{sk} \leftarrow \mathsf{MACGen}(1^{\lambda})$
$t \leftarrow \mathsf{MAC}(\mathsf{sk}, d_{f(x,z)})$
$v \leftarrow_{\$} \{0,1\}^{|t| + |d_{f(x,z)}|}$
$w := v \oplus (t || d_{f(x,z)})$
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 $i = 1, \dots, |z|$ do
    $\ell_i^{z_i} \leftarrow \mathsf{receiver}_{\mathsf{OT}}(z_i)$
$\mathcal{L}_z := \left\{ \ell_i^{z_i} \right\}_{i \in [|z|]}$
receive $_{\mathsf{O}}(k_j)$
for $i = 1, \dots, B$ do
    $\ell_{j,i}^{x_{j,i}} = \mathsf{PRF}(k_j, i)$
$\mathcal{L}_x := \left\{ \ell_{j,i}^{x_{j,i}} \right\}_{i \in [B]}$
receive $_{\mathsf{M}}(F, \mathsf{sk}, w)$
$Y \leftarrow \mathsf{Ev}\left(F, \mathcal{L}_x || \mathcal{L}_z\right)$
$Y_z || Y_{f(x,z)} := Y$
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 $\perp$
```

Fig. 3. METIS construction

a decoding information d_z and a random v. It returns (j, f, d_z, v) to $\mathcal{A}$. The adversary outputs some k'_j.

1. If for some i, $\mathsf{PRF}(k'_j, i) \neq \mathsf{PRF}(\mathsf{PRF}(k, j), i)$, then $\mathcal{S}_{\mathsf{O}}$ aborts by returning $\perp$ to $\mathcal{A}$.
2. Else, the adversary is provided with $Y_z = \left\{ \ell_i^{z_i} \right\}_{i \in [|z|]}$.

If $\mathcal{A}$ replies with $\perp$ then $\mathcal{S}_{\mathsf{O}}$ sends $\perp$ to $\mathcal{F}$. If it replies with v', then $\mathcal{S}_{\mathsf{O}}$ sends v' to $\mathcal{F}$ if $v' = v$ and aborts the simulation by outputting $\perp$ if $v' \neq v$. The inputs that $\mathcal{S}_{\mathsf{O}}$ provides to $\mathcal{A}$ are correctly distributed as in the real protocol. What is left to be shown is that the simulator aborts with the same probability as the real-world protocol. First we note that, $\mathcal{S}_{\mathsf{O}}$ checks whether the key k'_j produces the correct labels. Since the circuit is generated honestly, if there exists an i such that $\mathsf{PRF}(k'_j, i) \neq \mathsf{PRF}(\mathsf{PRF}(k, j), i)$, then the circuit cannot be evaluated. It follows that the real-world protocol also aborts. Due to the strong unforgeability of the one-time MAC scheme OTMAC the simulation aborts with the same probability (up to a negligible factor) when $v \neq v'$ in the real-world execution.

Corrupted Client. The simulator $\mathcal{S}_{\mathsf{C}}$ is described below. It makes use of the simulators $\mathcal{S}_{\mathsf{OT}}, \mathcal{S}_{\mathsf{ob}}$ and $\mathcal{S}_{\mathsf{prv}}$ that are guaranteed to exist for the security of oblivious transfer, obliviousness and privacy of the garbling scheme $\mathcal{G}$, respectively. On input (j, f) from the adversary $\mathcal{A}$, $\mathcal{S}_{\mathsf{C}}$ computes the keys k and k_j as specified in the protocol and generates $\mathcal{L}_x := \{\mathsf{PRF}(k_j, i)\}_{i \in [B]}$. It also sets $\mathcal{L}_z := \left\{ \ell_i^{z_i} \right\}_{i \in [|z|]}$ for random labels $\ell_i^{z_i} \leftarrow_{\$} \{0,1\}^{\lambda}$. It can now simulate the $|z|$ parallel oblivious transfer protocols for C using the simulator $\mathcal{S}_{\mathsf{OT}}$ and extract C's input z from $\mathcal{S}_{\mathsf{OT}}$. That is, in each iteration $\mathcal{S}_{\mathsf{OT}}$ outputs z_i and the simulator replies with $\ell_i^{z_i}$. Then $\mathcal{S}_{\mathsf{C}}$ can reconstruct z as $z_1 || \dots || z_{|z|}$. $\mathcal{S}_{\mathsf{C}}$ sends k_j to the adversary $\mathcal{A}$, then it sends (S_{ID}, z, f, j) to the ideal functionality where S_{ID} is an arbitrarily chosen session identifier. Depending on its reply we define two possible behaviours of the simulation:

1. If the response from $\mathcal{F}$ was $\perp$, then the simulator executes $F \leftarrow \mathcal{S}_{\mathsf{ob}}(1^{\lambda}, |z|, \mathcal{L}_x || \mathcal{L}_z)$. It generates a random w, a key $\mathsf{sk} \leftarrow \mathsf{MACGen}(1^{\lambda})$ and sends (F, sk, w) to $\mathcal{A}$. When it receives Y'_z from $\mathcal{A}$ it returns $\perp$ to $\mathcal{A}$.
2. If the response from $\mathcal{F}$ was some $y \neq \perp$, then the simulator obtains (F, d) using $\mathcal{S}_{\mathsf{prv}}(1^{\lambda}, |z|, y, \mathcal{L}_x || \mathcal{L}_z)$.

It generates w and sk as the previous case and sends (F, w, sk) to $\mathcal{A}$. When it receives Y'_z from $\mathcal{A}$, it checks if $z = \mathsf{De}(d_z, Y'_z)$, then returns $w \oplus \mathsf{MAC}(\mathsf{sk}, y)||y$ to $\mathcal{A}$. Else, the simulator aborts the execution.

The proof now follows a sequence of experiments. We define $\mathcal{S}_0$ as the real-world protocol, then we define the experiments $\mathcal{S}_1, \ldots, \mathcal{S}_{|z|}$ where we replace the first i oblivious transfers with the simulated protocol (denoted as $\mathcal{S}_{\mathsf{OT}}$). The indistinguishability $\mathcal{S}_0 \approx \mathcal{S}_1 \ldots \approx \mathcal{S}_{|z|}$ follows from the simulation based security of the oblivious transfer protocol.

In $\mathcal{S}_{|z|+1}$ we extract z from the outputs of $\mathcal{S}_{\mathsf{OT}}$ and abort the execution if this z is not obtained through decoding later. The simulations $\mathcal{S}_{|z|}$ and $\mathcal{S}_{|z|+1}$ differ only in the case that $\mathsf{De}(d_z, Y_{z'}) = z' \neq \perp$ and $z \neq z'$. Therefore we have that $\mathcal{S}_{|z|} \approx \mathcal{S}_{|z|+1}$, by the authenticity of the garbling scheme.

In $\mathcal{S}_{|z|+2}$ the garbled circuit is generated by $\mathcal{S}_{\mathsf{ob}}$, the simulator derived from the obliviousness of the garbling scheme, if z and f are not allowed by the policy of the owner. The indistinguishability follows from the the obliviousness of the garbling scheme.

In $\mathcal{S}_{|z|+3}$ the pair (F, d) is generated by $\mathcal{S}_{\mathsf{prv}}$, the simulator derived from the privacy of the garbling scheme, if z and f are allowed by the policy of the owner. It can be shown that $\mathcal{S}_{|z|+2} \approx \mathcal{S}_{|z|+3}$ holds by the privacy of the garbling scheme.

Finally, for $\mathcal{S}_{|z|+3} \approx \mathcal{S}_{\mathsf{C}}$ observe that the two executions are identical except that the decision whether to allow the computation of f on z is outsourced to $\mathcal{F}$. Therefore the two experiments are trivially indistinguishable to the eyes of the adversary.

Corrupted Server M**.** As before, the proof follows a sequence of experiments. We modify the real-world protocol through a series of experiments until we construct the simulator and we argue about the indistinguishability of the neighbouring simulations. Note that for the case of the corrupted server, we consider only a semi-honest adversary. We define $\mathcal{S}_0$ as the real-world protocol, then we substitute the i-th oblivious transfer with the simulated protocol in the experiments $\mathcal{S}_1, \ldots, \mathcal{S}_{|z|}$. In $\mathcal{S}_{|z|+1}$ we generate the block keys as random strings. For $1 \leq j \leq \left\lceil \frac{|x|}{B} \right\rceil$ we modify $\mathcal{S}_{|z|+1+j}$ such that the labels for the first j blocks are randomly generated. Finally $\mathcal{S}_{\mathsf{M}}$ is the simulator where the block index and function size are received from $\mathcal{F}$ as inputs.

The indistinguishability $\mathcal{S}_0 \approx \mathcal{S}_1 \approx \ldots \approx \mathcal{S}_{|z|}$ follows from the security of the oblivious transfer protocol. Given the pseudorandomness of the underlying PRF we can show that $\mathcal{S}_{|z|} \approx \mathcal{S}_{|z|+1}$ holds as the only difference between the simulations is that the PRFs are replaced with random strings. Similar to the case above, we can show $\mathcal{S}_{|z|+1} \approx \mathcal{S}_{|z|+2} \approx \ldots \approx \mathcal{S}_{|z|+1+\left\lceil \frac{|x|}{B} \right\rceil}$ by reducing it to the security of the underlying PRF. The indistinguishability between $\mathcal{S}_{|z|+1+\left\lceil \frac{|x|}{B} \right\rceil}$ and $\mathcal{S}_{\mathsf{M}}$ is trivial as there is no functional change between the games. □

4.4 Limitations

Our privacy model assumes that the information stored at the sequencing center is not leaked, i.e., we assume that the genomic information is securely erased after it is encoded and sent to the server M. We also assume, as discussed in Section 4.1, that the server is semi-honest and does not collude with the client. Actively corrupting both parties corresponds to having free access to the data owner's data regardless of his consent.

On a system level, our architecture does not directly defend against inference attacks by the clients. Although the data owner has full control over the functions that can be computed on the dataset, it is not always clear what an adversary can learn from the output of certain functions. Standard countermeasures against inference attacks, such as adding noise to achieve differential privacy [27], are compatible with our architecture but would penalize the performance of our system. Integrating differential privacy with our system without affecting efficiency is an interesting direction for future works.

5 Implementation

We now describe our prototype implementation of METIS. It is based on a modified version of Obliv-C [70], where the modifications are described in Appendix C. This allows us to use a well tested implementation for the most resource consuming part of the METIS protocol and only change Yao's protocol where our construction requires it: M plays the role of the garbler, using the precomputed labels, and client C acts as the evaluator of the garbled circuits. Our implementation of the sequencing center S is based on the programming language Python and we use the Python Cryptography Toolkit (pycrypto) [2] as this part is less relevant for the overall performance of the system.

5.1 Data Representation

For the evaluation of functions on the data owner's genomic data, the client needs to know the positions of specific genes in the encrypted data. This information is usually easily accessible through VCF files (as introduced in subsection 3.1). However, if one directly encrypts the VCF file, the encoding would leak the quantitative difference between the reference genome and the data owner's genome for individual genes as this difference is exactly what the VCF file encodes. A large difference in some gene strongly indicates a different gene product, which is a very sensitive information. Thus, we need to use an uncompressed representation and not just encrypt the VCF file. Developing such a representation is not trivial, because we have to use a fixed length encoding for every position of the reference genome while maintaining as much as possible of the expressiveness of the VCF file. For example, the VCF allows to encode insertions of arbitrary length. In order to obtain a fixed-length encoding, we need to set a suitable limit on this length.

In our implementation of METIS, the sequencing center S is encoding the following information: For every position i in the reference genome[1], we encrypt two bits (variant bits) that represent the genetic variant (None, SNP, Insertion, Deletion), b bits (length bits) that encode the length of an insertion/deletion and $2 \cdot (2^b - 1)$ bits (sequence bits) for the sequence of the insertion, where for the $2^b - 1$ possible positions we need two bits to encode the nucleotide base. Note that we have to do the encoding for both copies of the DNA (the maternal and the paternal copy) and always need all bits. Especially when there is no variant, we encrypt the variant, length and sequence bits which are all set to zero. This encoding clearly is constant-size for a fully sequenced genome.

5.2 Evaluation Functions

While METIS allows to compute every function on the DNA sequence, we restrict ourselves to evaluate the following functions, which are medically relevant.

SNP. *Is there an SNP at a specific position?*
AGL (4-Alpha-Glucanotransferas) is a gene in the glucose metabolism. Assuming a patient shows an unknown SNP at a specific position that has not been reported in any database, it is not clear whether the patient's disease is associated with this variant or not. Therefore it is relevant to find other people carrying this variant.

Heterozygous insertion. *Is there a heterozygous insertion at a specific position?*
Analogously, the patient's DNA sequence might show a heterozygous insertion (*i.e.* an insertion occurring only on the maternal or paternal copy) at a specific position of AGL. It is more difficult to determine whether there is a heterozygous insertion at a specific position than checking for an SNP because we have to also check whether the length and the sequence match. Furthermore, we have to ensure that the insertion only occurs on one copy.

Number of variants. *How many variants are there in a specific region?*
Besides looking for a specific variant, it is especially useful to check how many variants occur in a specific region. Assuming that diseases of the glucose metabolism are related to variants in specific regions of AGL, it is useful to detect patients who show a high number of variants in that region.

Frameshift. *Is there frameshift mutation in a specific region?*
The gene HCN2 (Hyperpolarization Activated Cyclic Nucleotide Gated Potassium And Sodium Channel 2) encodes an ion channel of the heart. Frameshift mutations are insertions and deletions of lengths that are not a multiple of three. In HCN2, a frameshift mutation cause the amino acid sequence of the ion channel to differ starting from the beginning of the frameshift. Thus, the channel might not function properly and cause heart diseases.

5.3 Optimizations

The implementation of free XOR [50] requires full control over the input labels. However, in the METIS system, the garbler only receives the fixed set of zero- and one-labels from the sequencing center S. Fortunately, for free XOR it is only required to fix an R consistent over all inputs such that $\ell^0_{j,i} = \ell^1_{j,i} \oplus R$ for all j and i. Therefore, in our implementation the sequencing center S samples a fresh R for each data owner and computes $\ell^{x_{j,i}}_{j,i} = \mathsf{PRF}(k_j, i)$ and $\ell^{1-x_{j,i}}_{j,i} = \ell^{x_{j,i}}_{j,i} \oplus R$, where we use HMAC with SHA256 as the PRF. As server M does not have access to k_j at any point in time, it can not distinguish whether the zero- or the one-label was derived from the PRF even when learning R. Given R, the server

[1] human genome assembly hg38 [5]

can now apply the free XOR optimization. Additionally, it is easy to see that M now only needs to store $\ell^0_{j,i}$ for all bits of the input as $\ell^1_{j,i}$ can be easily recomputed. The validity of this change follows naturally from the proof in [50]. We shall note that the free-XOR optimization assumes the existence of a random oracle.

Besides the free XOR optimization, Obliv-C provides the following optimizations: Half gates [71], point and permute [12], and garbled row reduction (GRR3) [60]. These optimizations directly apply to METIS since our modifications affect only the input labels.

6 Experimental Evaluation

While the METIS system has three parties interacting in the evaluation of a query, we focus on the interaction between the server and the client here as the data owner only sends small messages which are important for the security of the protocol but do not contribute to the computational feasibility of our construction. We ran both, the client and the server code, on the same machine for simplicity reasons. However, all communication between the client and the server still utilizes TCP network sockets in the same way as in a real-world implementation.

For the evaluation we used off-the-shelf hardware (Xeon Gold 6132, 2.60 GHz). Each party was assigned a single core. This is a reasonable assumption to make: For a real-world deployment we expect the query to be executed on many data owners' genetic information and therefore multiple independent instances of the protocol are executed in parallel.

Block Size. Both our construction and implementation are parameterized by the block size. The labels for one block are selected by a PRF using the same key derived from the master secret key. Selecting a "good" block size has several implications when applying the METIS system to real world problems:

- The communication cost of the data owner is roughly linear in the number of blocks processed. Thus, to keep the communication complexity as low as possible, one should try to keep the number of blocks low.
- The size of the garbled circuit executed between the client and the server is proportional to the sum of the size of all the touched blocks. When blocks are too large, a lot of unneeded information is processed by the protocol.
- If the client colludes with the server, they can obviously learn the data owner's data within the requested block(s). Data that was not part of any block where some colluding client was authorized to compute on, however, remains secure.

Block	$b = 2$	$b = 3$	$b = 4$	$b = 5$
exon	**0.074**	**0.141**	**0.259**	**0.497**
gene	**4.438**	**8.335**	**17.475**	**29.545**
chromosome	**124.357**	**215.987**	**389.657**	**751.498**

Table 1. Time of label generation (in seconds)

We suggest to choose blocks that correspond to biological units. Besides the trivial case of just having one block, we can choose each chromosome/gene/exon to be a block. The choice has performance and privacy implications. While having each exon to be a block is the best choice in case of performance, it is the worst in case of query privacy. On the other hand having only one block is the best choice for query privacy but the worst in terms of performance.

Setup Phase. In our genetic application, the setup phase is executed alongside the actual sequencing of a single data owner's genome at the sequencing center. Our naïve implementation took approximately 13 minutes to generate the labels of chromosome 1, which is the largest chromosome, for the garbled circuit using 5 bits for the length and thus 62 bits for the sequence (as discussed in subsection 5.1). The current implementation is saturating eight cores on our machine. The results for 2 to 5 bits are summarized in Table 1. Today's variant calling pipelines usually call short insertions or deletions with lengths up to 30. Thus $b = 5$ seems to be a practical choice. Note that the system can be easily adapted to changes of the pipelines by increasing b. As sequencing a full human genome still takes 26 hours [57], adding between half an hour ($b = 2$) and three hours ($b = 5$) for securing the data of the whole genome seems to be a modest overhead.

Evaluation Phase. Performance of METIS in the evaluation phase is more critical for the practicality of the scheme. We expect clients to process many queries on the data owner's DNA over time and each such query will likely process the genetic information of thousands of data owners – for example to find individuals suit-

	SNP		heterozygous insertion		number of variants		frameshift	
Block	**METIS**	**client**	**METIS**	**client**	**METIS**	**client**	**METIS**	**client**
exon	0.183	0.183	0.215	0.218	0.461	0.494		
gene	0.278	0.290	1.523	1.569	1.358	1.403	0.370	0.394
chromosome	107.842	107.925	1,507.182	1,507.232	1,346.115	1,346.464	264.999	265.189

Table 2. Time of function evaluation (in seconds for $b = 2$)

	SNP		heterozygous insertion		number of variants		frameshift	
Block	**METIS**	**client**	**METIS**	**client**	**METIS**	**client**	**METIS**	**client**
exon	0.183	0.183	0.226	0.232	0.566	0.612		
gene	0.281	0.294	2.099	2.145	1.369	1.415	0.409	0.445
chromosome	114.638	115.323	2,197.448	2,197.915	1,350.272	1,349.801	342.398	343.005

Table 3. Time of function evaluation (in seconds for $b = 3$)

	SNP		heterozygous insertion		number of variants		frameshift	
Block	**METIS**	**client**	**METIS**	**client**	**METIS**	**client**	**METIS**	**client**
exon	0.182	0.182	0.241	0.249	0.568	0.612		
gene	0.287	0.302	2.818	2.866	1.380	1.425	0.470	0.510
chromosome	125.764	126.716	2,962.965	2,962.378	1,361.069	1,361.620	426.638	428.050

Table 4. Time of function evaluation (in seconds for $b = 4$)

	SNP		heterozygous insertion		number of variants		frameshift	
Block	**METIS**	**client**	**METIS**	**client**	**METIS**	**client**	**METIS**	**client**
exon	0.183	0.184	0.259	0.270	0.570	0.616		
gene	0.326	0.345	3.749	3.795	1.393	1.441	0.539	0.588
chromosome	156.817	154.028	4,062.581	4,060.937	1,420.999	1,418.646	524.054	522.936

Table 5. Time of function evaluation (in seconds for $b = 5$)

	SNP			heterozygous insertion			number of variants			frameshift	
unit	**exon**	**gene**	**chr.**	**exon**	**gene**	**chr.**	**exon**	**gene**	**chr.**	**gene**	**chr.**
gates	$3.4 \cdot 10^3$	$2.7 \cdot 10^5$	$3.1 \cdot 10^8$	$2.1 \cdot 10^5$	$9.6 \cdot 10^6$	$2.2 \cdot 10^9$	$1.1 \cdot 10^6$	$3.3 \cdot 10^6$	$3.7 \cdot 10^9$	$9.2 \cdot 10^5$	$1.3 \cdot 10^9$
bytes	$2.3 \cdot 10^5$	$1.7 \cdot 10^7$	$2 \cdot 10^{10}$	$1.4 \cdot 10^7$	$6.2 \cdot 10^8$	$6.9 \cdot 10^{11}$	$7.1 \cdot 10^7$	$2.1 \cdot 10^8$	$2.4 \cdot 10^{11}$	$5.9 \cdot 10^7$	$8.1 \cdot 10^{10}$

Table 6. Circuit size (in number of gates) and network traffic (in bytes) for $b = 5$

able for clinical trials. We evaluated the server with the functions introduced in subsection 5.2 using the exon, gene (sum of all exons) or the chromosome (sum of all genes on that chromosome) as a block according to the above considerations about block sizes. The results of our experiments are summarized in Table 2 to Table 5 for $b = 2$ to $b = 5$ respectively. The number of gates of the circuits as well as the corresponding network traffic for the largest considered encoding ($b = 5$) are shown in Table 6. The circuit sizes increase when using larger blocks, as expected. In the computation of SNPs less bits are inspected compared to heterozygous insertions and the number of variants, which is also reflected in the circuit sizes. Using more bits of the encoding in the computation results in larger circuits. For the network traffic evaluation, we ran the experiment between two machines and used iptables accounting to extract the numbers. As one would expect, the traffic is dominated by the cost of transferring the circuit (as inputs are mostly calculated locally on the client side). We measured approximately 65 bytes per gate – close to what one would expect for a security level of 128 bits and four ciphertexts per gate.

In terms of computation cost, detecting whether there is a SNP on exon 14 of AGL when using the exon itself as a block can be processed (single-threaded) by the server in less than 200 ms. Assuming a powerful server with 48 Cores and 10,000 data owners, the protocol will be finished in slightly more than 40 seconds. If the client wants to hide the gene he is interested in, the entire chromosome 1 should be chosen as the block. Here, the detection still finishes in under two minutes and thus for 10,000 data owners in less than four hours. Note that the frameshift detection is over the entire gene HCN2 and thus there are no results for choosing a single exon as the block. Furthermore it seems that the frameshift detection is more efficient to perform than the SNP detection. However, this is because HCN2 is located on chromosome 19, which is considerably smaller than chromosome 1. In our encoding we encode an insertion with the bits 10 and a deletion with 11. Thus, in the first step of detecting a frameshift mutation, we only have to look at one of the variants bits to determine whether there is an insertion or deletion. The expensive step is testing whether the length of the insertion or deletion is a multiple of three. However, this step could be moved to the setup phase by spending one more bit in the encoding and then just checking this bit in the oblivious computation.

The detection of heterozygous insertions takes a lot longer than the other functions. The reason for this is that for every position of the block all bits of the encoding have to be processed. If the client is not interested in the sequence of the insertion but just in the position and the length, the evaluation time decreases significantly.

We did not evaluate our implementation against other systems suggested in subsection 1.3. This is due to the fast progress in development of MPC frameworks. Comparisons would not say anything about the actual systems but only about which system is built on top of the fastest framework.

7 Conclusions

We presented METIS, a service provider that supports evaluating functions over genetic data. In particular, METIS gives the data owner full control over the disclosure of information while splitting the computational burden between the server and the client. We showed that our construction is generically applicable to genetic studies and we demonstrated its practical feasibility by evaluating it on four medically relevant functions.

Acknowledgments

This work is a result of the collaborative research project PROMISE (16KIS0763 for FAU, 16KIS0364 for CeGaT) by the German Federal Ministry of Education and Research (BMBF). FAU authors were also supported by the German research foundation (DFG) through the collaborative research center 1223, and by the state of Bavaria at the Nuremberg Campus of Technology (NCT). NCT is a research cooperation between the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Technische Hochschule Nürnberg Georg Simon Ohm (THN).

References

[1] Breast cancer risk factors - genetics. http://www.breastcancer.org/risk/factors/genetics.

[2] Python cryptography toolkit (pycrypto). https://pypi.python.org/pypi/pycrypto. Accessed: 2017-05-18.

[3] Research – 23andme. https://www.23andme.com/en-int/research/. [Online; accessed 28-May-2018].

[4] Researchkit. http://researchkit.org/. [Online; accessed 28-May-2018].

[5] Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 02 2001.

[6] Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors. *ACM CCS 14*, Scottsdale, AZ, USA, November 3–7, 2014. ACM Press.

[7] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 535–548, Berlin, Germany, November 4–8, 2013. ACM Press.

[8] Erman Ayday, Emiliano De Cristofaro, Jean-Pierre Hubaux, and Gene Tsudik. Whole genome sequencing: Revolutionary medicine or privacy nightmare? *Computer*, 48(2):58–66, 2015.

[9] Erman Ayday, Jean Louis Raisaro, and Jean-Pierre Hubaux. Privacy-enhancing technologies for medical tests using genomic data. Technical report, 2012.

[10] Erman Ayday, Jean Louis Raisaro, Paul J McLaren, Jacques Fellay, and Jean-Pierre Hubaux. Privacy-preserving computation of disease risk by using genomic, clinical, and environmental data. In *HealthTech*, 2013.

[11] Pierre Baldi, Roberta Baronio, Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Countering gattaca: efficient and secure testing of fully-sequenced human genomes. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 691–702. ACM, 2011.

[12] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 503–513. ACM, 1990.

[13] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. Cryptology ePrint Archive, Report 2012/564, 2012. http://eprint.iacr.org/2012/564.

[14] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. Cryptology ePrint Archive, Report 2012/265, 2012. http://eprint.iacr.org/2012/265.

[15] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Yu et al. [69], pages 784–796.

[16] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[17] Ran Canetti and Juan A. Garay, editors. *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[18] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218, Dallas, TX, USA, May 23–26, 1998. ACM Press.

[19] Henry Carter, Charles Lever, and Patrick Traynor. Whitewash: Outsourcing garbled circuit generation for mobile devices. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 266–275. ACM, 2014.

[20] Henry Carter, Benjamin Mood, Patrick Traynor, and Kevin Butler. Secure outsourced garbled circuit evaluation for mobile devices. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, pages 289–304, Washington, D.C., 2013. USENIX.

[21] J.Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143 – 154, 1979.

[22] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. On the security of the Free-XOR technique. Cryptology ePrint Archive, Report 2011/510, 2011. http://eprint.iacr.org/2011/510.

[23] Peter JA Cock, Christopher J Fields, Naohisa Goto, Michael L Heuer, and Peter M Rice. The sanger fastq file format for sequences with quality scores, and the solexa/illumina fastq variants. *Nucleic acids research*, 38(6):1767–1771, 2010.

[24] Francis S Collins, Lisa D Brooks, and Aravinda Chakravarti. A dna polymorphism discovery resource for research on human genetic variation. *Genome research*, 8(12):1229–1231, 1998.

[25] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany.

[26] Petr Danecek, Adam Auton, Goncalo Abecasis, Cornelis A Albers, Eric Banks, Mark A DePristo, Robert E Handsaker, Gerton Lunter, Gabor T Marth, Stephen T Sherry, et al. The variant call format and vcftools. *Bioinformatics*, 27(15):2156–2158, 2011.

[27] Cynthia Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[28] Keith B Frikken. Practical private dna string searching and matching through efficient oblivious automata evaluation. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 81–94. Springer, 2009.

[29] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-Hop homomorphic encryption and rerandomizable Yao circuits. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 155–172, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

[30] Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Peter Rindal, and Mike Rosulek. Secure data exchange: A marketplace in the cloud. Cryptology ePrint Archive, Report 2016/620, 2016. http://eprint.iacr.org/2016/620.

[31] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[32] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*, volume 201, 2011.

[33] Yan Huang, Jonathan Katz, and David Evans. Quid-pro-quo-tocols: Strengthening semi-honest protocols with dual execution. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 272–284. IEEE, 2012.

[34] Yan Huang, Jonathan Katz, and David Evans. Efficient secure two-party computation using symmetric cut-and-choose. In Canetti and Garay [17], pages 18–35.

[35] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

[36] Thomas P Jakobsen, Jesper Buus Nielsen, and Claudio Orlandi. A framework for outsourcing of secure computation. In *Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security*, pages 81–92. ACM, 2014.

[37] Mark A Jensen, Vincent Ferretti, Robert L Grossman, and Louis M Staudt. The nci genomic data commons as an engine for precision medicine. *Blood*, 130(4):453–459, 2017.

[38] Somesh Jha, Louis Kruger, and Vitaly Shmatikov. Towards practical privacy for genomic computation. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 216–230. IEEE, 2008.

[39] Lynn B Jorde and Stephen P Wooding. Genetic variation, classification and'race'. *Nature genetics*, 36:S28–S33, 2004.

[40] Madhu Kalia. Personalized oncology: recent advances and future challenges. *Metabolism*, 62:S11–S14, 2013.

[41] Seny Kamara, Payman Mohassel, and Mariana Raykova. Outsourcing multi-party computation. *IACR Cryptology ePrint Archive*, 2011:272, 2011.

[42] Seny Kamara, Payman Mohassel, and Ben Riva. Salus: a system for server-aided secure function evaluation. In Yu et al. [69], pages 797–808.

[43] Murat Kantarcioglu, Wei Jiang, Ying Liu, and Bradley Malin. A cryptographic approach to securely share and query genomic sequences. *IEEE Transactions on information technology in biomedicine*, 12(5):606–617, 2008.

[44] Nikolaos Karvelas, Andreas Peter, Stefan Katzenbeisser, Erik Tews, and Kay Hamacher. Privacy-preserving whole genome sequence processing through proxy-aided oram. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, WPES '14, pages 1–10, New York, NY, USA, 2014. ACM.

[45] David J Kaufman, Juli Murphy-Bollinger, Joan Scott, and Kathy L Hudson. Public opinion about the importance of privacy in biobank research. *The American Journal of Human Genetics*, 85(5):643–654, 2009.

[46] Jane Kaye, Liam Curren, Nick Anderson, Kelly Edwards, Stephanie M Fullerton, Nadja Kanellopoulou, David Lund, Daniel G MacArthur, Deborah Mascalzoni, James Shepherd, et al. From patients to partners: participant-centric initiatives in biomedical research. *Nature Reviews Genetics*, 13(5):371, 2012.

[47] Miran Kim and Kristin Lauter. Private genome analysis through homomorphic encryption. Cryptology ePrint Archive, Report 2015/965, 2015. http://eprint.iacr.org/2015/965.

[48] Vladimir Kolesnikov, Payman Mohassel, and Mike Rosulek. FleXOR: Flexible garbling for XOR gates that beats free-XOR. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 440–457, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[49] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free xor gates and applications. *Automata, Languages and Programming*, pages 486–498, 2008.

[50] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 486–498, Reykjavik, Iceland, July 7–11, 2008. Springer, Heidelberg, Germany.

[51] Benjamin Kreuter, Abhi Shelat, Benjamin Mood, and Kevin RB Butler. Pcf: A portable circuit format for scalable two-party secure computation. In *Usenix Security*, volume 13, pages 321–336, 2013.

[52] Benjamin Kreuter, Abhi Shelat, and Chih-Hao Shen. Billion-gate secure computation with malicious adversaries. In *USENIX Security Symposium*, volume 12, pages 285–300, 2012.

[53] Yehuda Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. In Canetti and Garay [17], pages 1–17.

[54] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 36–54, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Heidelberg, Germany.

[55] Yehuda Lindell and Benny Pinkas. A proof of security of yao's protocol for two-party computation. *Journal of cryptology*, 22(2):161–188, 2009.

[56] Dahlia Malkhi, Noam Nisan, Benny Pinkas, Yaron Sella, et al. Fairplay-secure two-party computation system. In *USENIX Security Symposium*, volume 4. San Diego, CA, USA, 2004.

[57] Neil A. Miller, Emily G. Farrow, Margaret Gibson, Laurel K. Willig, Greyson Twist, Byunggil Yoo, Tyler Marrs, Shane Corder, Lisa Krivohlavek, Adam Walter, Josh E. Petrikin, Carol J. Saunders, Isabelle Thiffault, Sarah E. Soden, Laurie D. Smith, Darrell L. Dinwiddie, Suzanne Herd, Julie A. Cakici, Severine Catreux, Mike Ruehle, and Stephen F. Kingsmore. A 26-hour system of highly sensitive whole genome sequencing for emergency management of genetic diseases. *Genome Medicine*, 7(1):100, 2015.

[58] Benjamin Mood, Debayan Gupta, Kevin R. B. Butler, and Joan Feigenbaum. Reuse it or lose it: More efficient secure computation through reuse of encrypted values. In Ahn et al. [6], pages 582–596.

[59] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *31st ACM STOC*, pages 245–254, Atlanta, GA, USA, May 1–4, 1999. ACM Press.

[60] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *EC*, pages 129–139, 1999.

[61] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press.

[62] Muhammad Naveed, Shashank Agrawal, Manoj Prabhakaran, XiaoFeng Wang, Erman Ayday, Jean-Pierre Hubaux, and Carl A. Gunter. Controlled functional encryption. In Ahn et al. [6], pages 1280–1291.

[63] Boris Pasche and Devin Absher. Whole-genome sequencing: a step closer to personalized medicine. *JAMA*, 305(15):1596–1597, 2011.

[64] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

[65] Juan Ramón Troncoso-Pastoriza, Stefan Katzenbeisser, and Mehmet Celik. Privacy preserving error resilient dna searching through oblivious automata. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 519–528. ACM, 2007.

[66] Xiao Shaun Wang, Yan Huang, Yongan Zhao, Haixu Tang, XiaoFeng Wang, and Diyue Bu. Efficient genome-wide, privacy-preserving similar patient query based on private edit distance. In Indrajit Ray, Ninghui Li, and Christopher Kruegel:, editors, *ACM CCS 15*, pages 492–503, Denver, CO, USA, October 12–16, 2015. ACM Press.

[67] Mick Watson. Illuminating the future of dna sequencing. *Genome biology*, 15(2):108, 2014.

[68] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

[69] Ting Yu, George Danezis, and Virgil D. Gligor, editors. *ACM CCS 12*, Raleigh, NC, USA, October 16–18, 2012. ACM Press.

[70] Samee Zahur and David Evans. Obliv-c: A language for extensible data-oblivious computation. *IACR Cryptology ePrint Archive*, 2015:1153, 2015.

[71] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 220–250, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

A Cryptographic Building Blocks

We already described the cryptographic primitives we used to realize the METIS system and provide now their formal definition.

Definition A.1 (Pseudorandom function)**.** *Let* PRF : $\{0,1\}^\lambda \times \{0,1\}^{g(\lambda)} \rightarrow \{0,1\}^{h(\lambda)}$ *be an efficient, keyed function. We say that* PRF *is a* pseudorandom function *if for all probabilistic polynomial-time distinguishers D, there exists a negligible function* negl *such that:*

$$\left|\Pr\left[D^{\mathsf{PRF}_k(\cdot)}\left(1^\lambda\right)=1\right]-\Pr\left[D^{f(\cdot)}\left(1^\lambda\right)=1\right]\right|\le \mathsf{negl}(\lambda),$$

where $k \leftarrow \{0,1\}^\lambda$ is chosen uniformly at random and f is chosen uniformly at random from the set of functions mapping $g(\lambda)$-bit strings to $h(\lambda)$-bit strings.

Definition A.2 (Message authentication code)**.** *A* message authentication code *(MAC) is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{MACGen}, \mathsf{MAC}, \mathsf{MACVfy})$ *such that:*

1. *The* key generation algorithm MACGen *takes as input the security parameter 1^λ and outputs a key* sk *with* $|\mathsf{sk}| \ge \lambda$.
2. *The* tag generation algorithm MAC *takes as input a key* sk *and a message $m \in \{0,1\}^*$, and outputs a tag t.*
3. *The* verification algorithm MACVfy *takes as input a key* sk*, a message m, and a tag t. It outputs a bit b, with $b = 1$ meaning valid and $b = 0$ meaning invalid.*

For every λ, every key sk *output by* $\mathsf{MACGen}(1^\lambda)$*, and every $m \in \{0,1\}^*$, it holds that* $\mathsf{MACVfy}(\mathsf{sk}, m, \mathsf{MAC}(\mathsf{sk}, m)) = 1$.

Definition A.3 (Strongly secure one-time MAC)**.** *We define* strong security *for a one-time message authentication code* $\mathsf{OTMAC} = (\mathsf{MACGen}, \mathsf{MAC}, \mathsf{MACVfy})$ *by the experiment* $\mathsf{MAC}_{\mathcal{A},\mathsf{OTMAC}}(\lambda)$ *shown in Figure 4.*

```
$\mathsf{MAC}_{\mathcal{A},\mathsf{OTMAC}}(\lambda)$
$\mathsf{sk} \leftarrow \mathsf{MACGen}(1^\lambda)$
$(m,t) \leftarrow \mathcal{A}^{\mathsf{MAC}'(\mathsf{sk},\cdot)}(1^\lambda)$
if $\mathsf{MACVfy}(\mathsf{sk},m,t) = 1$ and $(m,t) \notin \mathcal{Q}$
    return 1

Oracle $\mathsf{MAC}'(\mathsf{sk},m)$
if $\mathcal{Q} \neq \emptyset$ abort
$t \leftarrow \mathsf{MAC}(\mathsf{sk},m)$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup (m,t)$
return $t$
```

Fig. 4. Message authentication experiment.

OTMAC *is strongly secure, if for all* PPT *adversaries $\mathcal{A}$, there is a negligible function* negl *such that*

$$\Pr\left[\mathsf{MAC}_{\mathcal{A},\mathsf{OTMAC}}(\lambda)=1\right]\le \mathsf{negl}(\lambda).$$

Definition A.4 (Oblivious transfer [64])**.** *An* oblivious transfer *(OT) is a two-party functionality $\mathcal{F}_{\mathsf{OT}}$, involving a sender $\mathcal{S}$ with input x_0, x_1 and a receiver $\mathcal{R}$ with input $b \in \{0,1\}$ as follows:*

1. *Upon receiving a message $(S_{ID}, \mathsf{sender}_{\mathsf{OT}}, x_0, x_1)$ from $\mathcal{S}$, where each $x_i \in \{0,1\}^\ell$, store (x_0, x_1) (The lengths of the strings ℓ is fixed and known to all parties).*
2. *Upon receiving a message $(S_{ID}, \mathsf{receiver}_{\mathsf{OT}}, b)$ from $\mathcal{R}$, check if a $(S_{ID}, \mathsf{sender}_{\mathsf{OT}}, \dots)$ message was previously sent. If yes, send (S_{ID}, x_b) to $\mathcal{R}$ and (S_{ID}) to the adversary $\mathcal{S}$ and halt. If not, send nothing to $\mathcal{R}$ (but continue running).*

For a garbling scheme we use the notation and definitions of [15] with minor modifications. Specifically, we allow one to fix the encoding X for a certain input x and to sample a fresh encoding e under the constraint that $X \leftarrow \mathsf{En}(e, x)$. This can be easily done for Yao's [68] scheme by fixing the labels $\{\ell^{x_i}\}_{i\in\lambda}$ and randomly sampling the remaining ones.

Definition A.5 (Garbling scheme)**.** *A* garbling scheme $\mathcal{G} = (\mathsf{GenEn}, \mathsf{Gb}, \mathsf{En}, \mathsf{ComEn}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ *is a tuple of algorithms, where the string f, the original function, describes the function* $\mathsf{ev}(f,\cdot) : \{0,1\}^{\ell(x)} \to \{0,1\}^{\ell(y)}$ *that we want to garble. The remaining algorithms are defined as:*

1. *The probabilistic encoding generation algorithm* GenEn *takes as input the security parameter 1^λ and outputs the encoding information e.*
2. *The probabilistic garbling algorithm* Gb *takes as input the security parameter 1^λ, a function f and an encoding information e. It outputs a garbled function F and the decoding information d.*
3. *The deterministic encoding algorithm* En *takes as input the encoding information e and an initial input $x \in \{0,1\}^{\ell(x)}$. It outputs a garbled input $\mathcal{L}$.*
4. *The probabilistic complement encoding algorithm* ComEn *takes as input the initial input x and the garbled input $\mathcal{L}$. It outputs an encoding information e, such that* $\mathcal{L} = \mathsf{En}(x, e)$.
5. *The deterministic evaluation algorithm* Ev *takes as input a garbled function F and a garbled input $\mathcal{L}$ to produce a garbled output Y.*
6. *The deterministic* decoding algorithm De *takes as input the decoding information d and a garbled output Y. It outputs a final output $y \in \{0,1\}^{\ell(y)}$.*

We assume our encoding algorithm to be projective.

Definition A.6 (Projective garbling scheme). *A garbling scheme* $\mathcal{G} = (\mathsf{GenEn}, \mathsf{Gb}, \mathsf{En}, \mathsf{ComEn}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ *is* projective, *if for all* $f, x, x' \in \{0,1\}^{\ell(x)}$, $\lambda \in \mathbb{N}$, $(F, d) \leftarrow \mathsf{Gb}(1^\lambda, f, e)$, $\mathcal{L} = \mathsf{En}(e, x)$ *and* $\mathcal{L}' = \mathsf{En}(e, x')$, *it holds that* $\mathcal{L} = (\mathcal{L}_1, \dots, \mathcal{L}_{\ell(x)})$ *and* $\mathcal{L}' = (X'_1, \dots \mathcal{L}'_{\ell(x)})$ *with* $|\mathcal{L}_i| = |\mathcal{L}'_i|$, *and* $\mathcal{L}_i = \mathcal{L}'_i$ *whenever* $x_i = x'_i$.

In the following we recall the security properties of a garbling scheme. Our notions follow as an adaptation of the properties defined by Bellare et al. [15] with the only difference being that the adversary can fix the encoding of its input. The standard scheme from Yao [68] is shown to satisfy the standard notions of privacy, authenticity, and obliviousness [15] and can be proven secure against the properties defined below with a similar argument.

Definition A.7 (Simulation privacy of a garbling scheme). *Let* $\mathcal{G} = (\mathsf{GenEn}, \mathsf{Gb}, \mathsf{En}, \mathsf{ComEn}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ *be a garbling scheme and* $\Phi()$ *a side information function. We define* privacy *in the game* $\mathrm{PrvSim}_{\mathcal{G},\Phi(),\mathsf{Sim}}$ *shown in Figure 5.*

```
$\mathrm{PrvSim}_{\mathcal{G},\Phi(),\mathsf{Sim}}$
$(f, x_1, \mathcal{L}_1, x_2) \leftarrow \mathcal{A}(1^\lambda)$
if $x_1||x_2 \notin \{0,1\}^{\ell(x)}$ return $\perp$
$e_1 \leftarrow \mathsf{ComEn}(x_1, \mathcal{L}_1)$
$e_2 \leftarrow\$ \{0,1\}^{|x_2| \cdot 2\lambda}$
$\mathcal{L}_2 \leftarrow \mathsf{En}(e_2, x_2)$
$b \leftarrow\$ \{0,1\}$
if $b = 0$ then
    $(F, d) \leftarrow \mathsf{Gb}(1^\lambda, f, e_1||e_2)$
if $b = 1$ then
    $(F, d) \leftarrow \mathsf{Sim}(1^\lambda, \Phi(f), f(x), \mathcal{L}_1||\mathcal{L}_2)$
$b' \leftarrow \mathcal{A}(F, \mathcal{L}, d)$
return $b = b'$
```

Fig. 5. Simulation privacy game.

The advantage of adversary $\mathcal{A}$ *in game* $\mathrm{PrvSim}_{\mathcal{G},\Phi(),\mathsf{Sim}}$ *is defined as:*

$$\mathsf{Adv}^{\mathrm{prv}}_{\mathcal{G},\Phi(),\mathsf{Sim},\mathcal{A}}(\lambda) = 2\Pr[\mathrm{PrvSim}_{\mathcal{G},\Phi(),\mathsf{Sim}}(\mathcal{A}, \lambda)] - 1.$$

$\mathcal{G}$ *is prv.sim secure over* $\Phi()$ *if for all PPT adversaries* $\mathcal{A}$, *there exists a PPT simulator* Sim, *such that* $\mathsf{Adv}^{\mathrm{prv}}_{\mathcal{G},\Phi(),\mathsf{Sim},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$.

Definition A.8 (Authenticity of a garbling scheme). *Let* $\mathcal{G} = (\mathsf{GenEn}, \mathsf{Gb}, \mathsf{En}, \mathsf{ComEn}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ *be a garbling scheme. We define* authenticity *in the game* $\mathrm{Aut}^{\mathcal{A}}_{\mathcal{G}}$ *shown in Figure 6.*

```
$\mathrm{Aut}^{\mathcal{A}}_{\mathcal{G}}$
$(f, x_1, \mathcal{L}_1, x_2) \leftarrow \mathcal{A}(1^\lambda)$
if $x_1||x_2 \notin \{0,1\}^{\ell(x)}$ return $\perp$
$e_1 \leftarrow \mathsf{ComEn}(x_1, \mathcal{L}_1)$
$e_2 \leftarrow\$ \{0,1\}^{|x_2| \cdot 2\lambda}$
$\mathcal{L}_2 \leftarrow \mathsf{En}(e_2, x_2)$
$(F, d) \leftarrow \mathsf{Gb}(1^\lambda, f, e_1||e_2)$
$Y \leftarrow \mathcal{A}(F, \mathcal{L}_2)$
return $(Y \neq \mathsf{Ev}(F, \mathcal{L}_1||\mathcal{L}_2)$ and $\mathsf{De}(d, Y) \neq \perp)$
```

Fig. 6. Authenticity game.

The advantage of an adversary $\mathcal{A}$ *in game* $\mathrm{Aut}^{\mathcal{A}}_{\mathcal{G}}$ *is defined as:*

$$\mathsf{Adv}^{\mathrm{aut}}_{\mathcal{G}}(\lambda) = \Pr[\mathrm{Aut}^{\mathcal{A}}_{\mathcal{G}}(\lambda)].$$

A garbling scheme $\mathcal{G}$ *provides authenticity if the advantage* $\mathsf{Adv}^{\mathrm{aut}}_{\mathcal{G}}(\lambda)$ *is* $\mathsf{negl}(\lambda)$ *for all* PPT *adversaries* $\mathcal{A}$.

Definition A.9 (Obliviousness of a garbling scheme). *Let* $\mathcal{G} = (\mathsf{GenEn}, \mathsf{Gb}, \mathsf{En}, \mathsf{ComEn}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ *be a garbling scheme. We define* obliviousness *in the game* $\mathrm{ObvSim}_{\mathcal{G},\mathsf{Sim}}$ *shown in Figure 7.*

```
$\mathrm{ObvSim}_{\mathcal{G},\mathsf{Sim}}$
$(f, x_1, \mathcal{L}_1, x_2) \leftarrow \mathcal{A}(1^\lambda)$
if $x_1||x_2 \notin \{0,1\}^{\ell(x)}$ return $\perp$
$e_1 \leftarrow \mathsf{ComEn}(x_1, \mathcal{L}_1)$
$e_2 \leftarrow\$ \{0,1\}^{|x_2| \cdot 2\lambda}$
$\mathcal{L}_2 \leftarrow \mathsf{En}(e_2, x_2)$
$b \leftarrow\$ \{0,1\}$
if $b = 0$ then
    $(F, d) \leftarrow \mathsf{Gb}(1^\lambda, f, e_1||e_2)$
if $b = 1$ then
    $F \leftarrow \mathsf{Sim}(\Phi(f), \mathcal{L}_1||\mathcal{L}_2)$
$b' \leftarrow \mathcal{A}(F, \mathcal{L})$
return $b = b'$
```

Fig. 7. Simulation obliviousness game.

The advantage of an adversary $\mathcal{A}$ *in game* $\mathrm{ObvSim}_{\mathcal{G},\mathsf{Sim}}$ *is defined as:*

$$\mathsf{Adv}^{\mathrm{aut}}_{\mathcal{G}}(\lambda) = \Pr[\mathrm{ObvSim}_{\mathcal{G},\mathsf{Sim}}(\lambda)].$$

A garbling scheme $\mathcal{G}$ *provides obliviousness if the advantage* $\mathsf{Adv}^{\mathrm{aut}}_{\mathcal{G}}(\lambda)$ *is* $\mathsf{negl}(\lambda)$ *for all* PPT *adversaries* $\mathcal{A}$.

B Security Analysis

Proof of Theorem 4.2. We show that construction 3 is secure by considering separately each party involved in the protocol. For each case we build a simulator that interacts with the ideal functionality while simulating the complete view of an execution for an adversary that has corrupted the party in the real world. For each of the corrupted parties, we first provide the reader with the description of the simulator and then we argue why the simulation is consistent with the view of the adversary in the real world protocol. The formal argument follows.

Corrupted Data Owner. Consider the first case where the data owner O is corrupted by an adversary $\mathcal{A}$. The corresponding simulator is described in the following.

Description of $\mathcal{S}_{\mathsf{O}}$:

1. Simulator $\mathcal{S}_{\mathsf{O}}$ receives a handler, a query and an index of the requested block denoted by (h, z, f, j) from $\mathcal{F}$ as input.
2. $\mathcal{S}$ samples a random $k \leftarrow_{\$} \{0,1\}^{\lambda}$ and sends it to the adversary $\mathcal{A}$.
3. The simulator samples $|z|$-many pairs of random labels $\left\{(\ell_i^0, \ell_i^1)\right\}_{i \in [|z|]}$. Let d_z be the decoding information of those labels. $\mathcal{S}_{\mathsf{O}}$ then samples a random $v \leftarrow_{\$} \{0,1\}^{|t|+|d_{f(x,z)}|}$, where $|t|$ denotes the size of a one-time MAC t. The simulator finally sends (j, f, d_z, v) to $\mathcal{A}$.
4. The adversary outputs some k'_j and $\mathcal{S}_{\mathsf{O}}$ checks whether there is an $i \in [B]$ such that $\mathsf{PRF}(k'_j, i) \neq \mathsf{PRF}(\mathsf{PRF}(k, j), i)$. If the condition holds then $\mathcal{S}_{\mathsf{O}}$ aborts the simulation by returning $\perp$ to $\mathcal{A}$. Otherwise the adversary is provided with $Y_z = \left\{\ell_i^{z_i}\right\}_{i \in [|z|]}$.
5. If $\mathcal{A}$ replies with $\perp$ then $\mathcal{S}_{\mathsf{O}}$ sends $\perp$ to $\mathcal{F}$. If it replies with v', then $\mathcal{S}_{\mathsf{O}}$ sends v' to $\mathcal{F}$ if $v' = v$ and aborts the simulation by outputting $\perp$ if $v' \neq v$.

Analysis: The simulator clearly runs in polynomial time. Also it is easy to see that the inputs that $\mathcal{S}_{\mathsf{O}}$ provides to $\mathcal{A}$ are correctly distributed, since d_z is the decoding information of random output labels, as in the real protocol. What we need to show is that any attempt to deviate from the honest execution of the protocol by $\mathcal{A}$ is reflected by the simulator with an abort in the ideal world. First we note that in the third step of the simulation, $\mathcal{S}_{\mathsf{O}}$ checks whether the key k'_j produces the correct labels. Since the circuit is generated honestly, if there exists an i such that $\mathsf{PRF}(k'_j, i) \neq \mathsf{PRF}(\mathsf{PRF}(k, j), i)$, then the circuit cannot be evaluated. It follows that the abort of the simulator corresponds to the abort in the real protocol. We now have to show that the same holds for the abort in the fifth step. Obviously, if $\mathcal{A}$ outputs $\perp$, both the simulation and the real world protocol abort with probability 1. On the other hand, the simulation aborts with the same probability when $v \neq v'$, while it is in principle not clear whether the same happens in the real-world execution. Therefore we have to prove that $\Pr[\perp_{\mathsf{real}} | v \neq v'] \geq 1 - \mathsf{negl}(\lambda)$, for some negligible function $\mathsf{negl}(\lambda)$. Assume towards contradiction that $\Pr[\perp_{\mathsf{real}} | v \neq v'] \leq 1 - \epsilon(\lambda)$, for some non-negligible function $\epsilon(\lambda)$. Then we can construct the following reduction against the strong unforgeability of the one-time MAC scheme OTMAC.

> $\underline{\mathcal{R}(1^{\lambda})}$: $\mathcal{R}$ chooses an arbitrary z and samples the decoding tables d_z and $d_{f(x,z)}$, as specified by the original protocol. It queries its OTMAC oracle for a MAC on $d_{f(x,z)}$ and obtains t. It then behaves exactly like the above simulator $\mathcal{S}$ in generating messages and sending messages except that it now had set $v \leftarrow_{\$} \{0,1\}^{|t|+|d_{f(x,z)}|}$ and $w = v \oplus (t || d_{f(x,z)})$. Adversary $\mathcal{A}$ now replies with some v'. $\mathcal{R}$ computes $w \oplus v'$ to obtain $t' || d'_{f(x,z)}$. Reduction $\mathcal{R}$ simply returns $(t', d'_{f(x,z)})$ to the challenger.

The reduction is clearly efficient and the inputs provided to the adversary are correctly distributed (since it follows in verbatim the real world execution). By initial assumption we have that $\Pr[\perp_{\mathsf{real}} | v \neq v'] \leq (1 - \epsilon(\lambda))$, which implies that with non-negligible probability $\epsilon(\lambda)$ the real world protocol does not abort and that $v' \neq v$. It follows that $\mathsf{MACVfy}(\mathsf{sk}, d'_{f(x,z)}, t') = 1$ and that $(t' || d'_{f(x,z)}) = w \oplus v' \neq w \oplus v = (t || d_{f(x,z)})$. We can conclude that either $t \neq t'$ or $d_{f(x,z)} \neq d'_{f(x,z)}$ with probability $\epsilon(\lambda)$, which is a contradiction to the strong unforgeability of the one-time MAC scheme OTMAC and concludes our proof.

Semi-honest Server M. We now consider the case of the server M being passively corrupted by an adversary $\mathcal{A}$. Let $\mathcal{S}_{\mathsf{OT}}$ be the simulator that is guaranteed to exist for M in the oblivious transfer protocol. The proof outline is as follows:

$\mathcal{S}_0$ resembles exactly the real world protocol.
$\mathcal{S}_i$ for $1 \leq i \leq |z|$ is defined as $\mathcal{S}_0$ except the first i oblivious transfers are simulated by $\mathcal{S}_{\mathsf{OT}}$.

$\mathcal{S}_{|z|+1}$ is defined as $\mathcal{S}_{|z|}$ except that the keys for each block (k_j's) of x are generated randomly instead of the $\mathsf{PRF}(k,\cdot)$.

$\mathcal{S}_{|z|+1+j}$ for $1 \leq j \leq \left\lceil \frac{|x|}{B} \right\rceil$ is defined as $\mathcal{S}_{|z|+1}$ except that both labels for bits in the first j blocks are generated randomly (a truly random string replaces the $\mathsf{PRF}(k_j,\cdot)$ which was used for one of the two labels for each bit).

$\mathcal{S}_{\mathsf{IDEAL}}$ is defined as $\mathcal{S}_{|z|+1+\left\lceil \frac{|x|}{B} \right\rceil}$ except that it receives block indices, the query size and the function as input from the ideal functionality $\mathcal{F}$.

$\mathcal{S}_0 \approx \mathcal{S}_1 \approx \ldots \approx \mathcal{S}_{|z|}$: Simulators are defined as the real world protocol except that in the simulator $\mathcal{S}_i$ the first i oblivious transfers is simulated through $\mathcal{S}_{\mathsf{OT}}$. Assuming towards contradiction that there exists an adversary $\mathcal{A}$ that can distinguish any two consecutive simulations with some non-negligible probability $\epsilon(\lambda)$. Since the only difference between the two simulations is one of the oblivious transfer simulation, by the security of the simulator $\mathcal{S}_{\mathsf{OT}}$, such an $\mathcal{A}$ cannot exist.

$\mathcal{S}_{|z|} \approx \mathcal{S}_{|z|+1}$: In the simulator $\mathcal{S}_{|z|+1}$ the keys k_j's for $1 \leq j \leq \left\lceil \frac{|x|}{B} \right\rceil$ are generated randomly instead of using $k_j \leftarrow \mathsf{PRF}(k,j)$ in the simulator $\mathcal{S}_{|z|}$. This is the only difference between the simulators. Assuming towards contradiction that there exists an adversary $\mathcal{A}$ that can distinguish the simulations output by these simulators with a non-negligible probability $\epsilon(\lambda)$, we build a reduction $\mathcal{R}$ against the security of the underlying PRF.

<u>$\mathcal{R}(1^\lambda)$:</u> $\mathcal{R}$ has access to its oracle that is either a $\mathsf{PRF}(k,\cdot)$ $(b=0)$ or a truly random function $(b=1)$. The reduction $\mathcal{R}$ starts by picking a random j and $|z|$ and queries its own oracle for $j = 1, \ldots, \left\lceil \frac{|x|}{B} \right\rceil$ (which are the block indices) to obtain $k_1, \ldots, k_{\left\lceil \frac{|x|}{B} \right\rceil}$. It then generates the labels for each bit position of x using the keys obtained just as in the real protocol. From now on, it constructs the simulation transcript just as the simulators (note that from now both $\mathcal{S}_{|z|}$ and $\mathcal{S}_{|z|+1}$ behave the same way), and returns the transcript to $\mathcal{A}$. Adversary $\mathcal{A}$ now replies with a bit b' which the reduction replies as its own.

We observe that the reduction is efficient. Note that if $b = 0$ then the reduction perfectly simulates the experiment $\mathcal{S}_{|z|}$, while if $b = 1$ then the transcript of the reduction is the same as $\mathcal{S}_{|z|+1}$. Therefore we can say that $\Pr[\mathcal{A} \text{ wins}] = \Pr[\mathcal{R} \text{ wins}] \geq \epsilon(\lambda)$. This is a contradiction to the security of the PRF.

$\mathcal{S}_{|z|+1} \approx \mathcal{S}_{|z|+2} \approx \ldots \approx \mathcal{S}_{|z|+1+\left\lceil \frac{|x|}{B} \right\rceil}$: For $1 \leq j \leq \left\lceil \frac{|x|}{B} \right\rceil$, in simulator $\mathcal{S}_{|z|+1+j}$, both labels for bits in the first j blocks are generated randomly. Therefore each of the simulators differ from each other only in replacing $\mathsf{PRF}(k_j, i)$ for $i = 1, \ldots |B|$ by truly random strings (computes e_x by generating random labels $\ell_{j,i}^{x_{j,i}}, \ell_{j,i}^{1-x_{j,i}} \leftarrow_{\$} \{0,1\}^\lambda$ for $i = 1, \ldots, |B|$ where $|B|$ is the block size). Given that $\mathsf{PRF}(\cdot,\cdot)$ is a secure PRF, we can show (with an argument similar as above) that the simulations are indistinguishable from each other.

$\mathcal{S}_{|z|+1+\left\lceil \frac{|x|}{B} \right\rceil} \approx \mathcal{S}_{\mathsf{IDEAL}}$: The simulator $\mathcal{S}_{\mathsf{IDEAL}}$ computes e_x by generating random labels $\ell_{j,i}^{x_{j,i}}, \ell_{j,i}^{1-x_{j,i}} \leftarrow_{\$} \{0,1\}^\lambda$ for $i = 1, \ldots, |B|$ where $|B|$ is the block size. Upon receiving j from the ideal functionality it executes the protocol as specified in the scheme and provides $\mathcal{A}$ with the transcript of the run. Since the change from $\mathcal{S}_{|z|+1+\left\lceil \frac{|x|}{B} \right\rceil}$ is only conceptual, the equality trivially holds.

Note that $\mathcal{S}_{\mathsf{IDEAL}}$ interacts only with $\mathcal{F}$, the ideal functionality. From the above equalities we see that, through transitivity, $\mathcal{S}_0 \approx \mathcal{S}_1 \approx \ldots \approx \mathcal{S}_{|z|} \approx \mathcal{S}_{\mathsf{IDEAL}}$. This proves the view of the adversary corrupting M semi-honestly in the real world execution is computationally indistinguishable to its view in the simulated execution.

Corrupted Client. Consider the case where the client C is corrupted by an adversary $\mathcal{A}$. Let $\mathcal{S}_{\mathsf{OT}}$ be the simulator that is guaranteed to exist for C in the oblivious transfer protocol. Let $\mathcal{S}_{\mathsf{prv}}$ and $\mathcal{S}_{\mathsf{ob}}$ be the simulators that are guaranteed to exist by the privacy and obliviousness of the garbling scheme $\mathcal{G}$. The proof outline is as follows:

$\mathcal{S}_0$ resembles exactly the real world protocol.

For $1 \leq i \leq |z|$, $\mathcal{S}_i$ is defined as $\mathcal{S}_0$ except the first i oblivious transfers are simulated by $\mathcal{S}_{\mathsf{OT}}$.

$\mathcal{S}_{|z|+1}$ is defined to be the same as $\mathcal{S}_{|z|}$ except that z is extracted from simulators of the oblivious transfer. Then the simulator returns the correct decoding information $d_{f(x,z)}$ to the adversary if z and f satisfy the policy specified by the owner. At some point of the execution the adversary returns some $Y_{z'}$, the simulator checks whether $z \neq \mathsf{De}(d_z, Y_{z'})$ and $\mathsf{De}(d_z, Y_{z'}) \neq \perp$ and it aborts the execution if this is the case.

$\mathcal{S}_{|z|+2}$ is defined as $\mathcal{S}_{|z|+1}$ except that simulator $\mathcal{S}_{\mathsf{ob}}$ is used to generate F, if z and f are not allowed by the policy of the owner. On input $Y_{z'}$ by $\mathcal{A}$ the simulator simply replies with $\perp$.

$\mathcal{S}_{|z|+3}$ is defined as $\mathcal{S}_{|z|+2}$ except that simulator $\mathcal{S}_{\mathsf{prv}}$ is used to generate (F, d) if z and f are allowed by the

policy of the owner.
$\mathcal{S}_{\mathsf{C}}$ is defined as $\mathcal{S}_{|z|+3}$ except that it executes the protocol in the ideal world: The extracted function z is sent to the ideal functionality along with S_{ID}, f, j and the next message of the simulation depends on whether $\mathcal{F}$ returns $\perp$ or some $f(x, z)$.

The full description of the simulator is presented in the following. *Description of* $\mathcal{S}_{\mathsf{C}}$*:*

1. $\mathcal{S}_{\mathsf{C}}$ receives a block index j and function f from the adversary $\mathcal{A}$.
2. $\mathcal{S}_{\mathsf{C}}$ computes the key k as specified in the original protocol then it sets $k_j \leftarrow \mathsf{PRF}(k, j)$ and $\mathcal{L}_x := \{\mathsf{PRF}(k_j, i)\}_{i \in [B]}$.
3. $\mathcal{S}_{\mathsf{C}}$ computes for all $i \in \{1, \ldots, |z|\}$ a random label $\ell_i \leftarrow \{0,1\}^\lambda$ and sets $\mathcal{L}_z := \left\{\ell_i^{z_i}\right\}_{i \in [|z|]}$.
4. $\mathcal{S}_{\mathsf{C}}$ simulates the $|z|$-many parallel oblivious transfer protocols for C using the simulator $\mathcal{S}_{\mathsf{OT}}$. In each iteration $\mathcal{S}_{\mathsf{OT}}$ outputs z_i and the simulator replies with ℓ_i. The simulator can reconstruct z as $z_1 \| \ldots \| z_{|z|}$.
5. $\mathcal{S}_{\mathsf{C}}$ sends k_j to the adversary $\mathcal{A}$ on behalf of the data owner O, then it sends (S_{ID}, z, f, j) to the ideal functionality, which replies with either $\perp$ or some y. Here S_{ID} is an arbitrarily chosen session identifier. We distinguish the two cases defining two possible behaviours of the simulation.
6. If the response from $\mathcal{F}$ was $\perp$, then the simulator does the following:
   (a) It executes $F \leftarrow \mathcal{S}_{\mathsf{ob}}(1^\lambda, |Z|, \mathcal{L}_x || \mathcal{L}_z)$.
   (b) It generates a random w of the appropriate length.
   (c) It generates, a key $\mathsf{sk} \leftarrow \mathsf{MACGen}(1^\lambda)$.
   (d) It sends (F, sk, w) to $\mathcal{A}$.
   (e) It then receives Y'_z from $\mathcal{A}$. It returns $\perp$ to $\mathcal{A}$.
7. If the response from $\mathcal{F}$ was some $y \neq \perp$, then the simulator behaves as follows:
   (a) It executes $(d, F) \leftarrow \mathcal{S}_{\mathsf{prv}}(1^\lambda, |z|, y, \mathcal{L}_x || \mathcal{L}_z)$ and parses $d = d_z || d_y$.
   (b) It generates a random w of the appropriate length.
   (c) It generates, a key $\mathsf{sk} \leftarrow \mathsf{MACGen}(1^\lambda)$.
   (d) It sends (F, sk, w) to $\mathcal{A}$.
   (e) It then receives Y'_z from $\mathcal{A}$. It computes $z' = \mathsf{De}(d_z, Y'_z)$ and checks whether $z = z'$. If the check holds then it replies with $w \oplus \mathsf{MAC}(\mathsf{sk}, y) || y$ to $\mathcal{A}$. If $z' \neq z$ then the simulator aborts the execution.

$\mathcal{S}_0 \approx \mathcal{S}_1 \ldots \approx \mathcal{S}_{|z|}$: The simulators are defined to behave as in the real world protocol specification except that in the simulator $\mathcal{S}_i$ the first i oblivious transfers are simulated through $\mathcal{S}_{\mathsf{OT}}$. Since the difference between each neighbouring pair of simulations is in the usage of $\mathcal{S}_{\mathsf{OT}}$, we can prove indistinguishability by a standard hybrid argument.

$\mathcal{S}_{|z|} \approx \mathcal{S}_{|z|+1}$: We note that the two simulations are identical except that in $\mathcal{S}_{|z|}$ the decision of whether sending d_y or not is taken basing on the value of $\mathsf{De}(d_z, Y_{z'})$, while in $\mathcal{S}_{|z|+1}$ the decision is taken basing on the extracted z. Clearly the two simulations differ only in the case that $\mathsf{De}(d_z, Y_{z'}) = z' \neq \perp$ and $z \neq z'$. Therefore to prove that the two simulations are indistinguishable it is enough to show that the probability of the simulation to abort is bounded from above by a negligible function. Assume towards contradiction that the simulation aborts with probability $\epsilon(\lambda)$, for some non-negligible function ϵ. Then we can construct the following reduction against the authenticity of the garbling scheme.

> $\underline{\mathcal{R}(1^\lambda)}$: $\mathcal{R}$ simulates the experiment as specified in $\mathcal{S}_{|z|}$ until the simulation of the parallel OTs where it extracts the function z, then it sends to its challenger the tuple $(\nabla, x, \mathcal{L}_x, z)$. As a response it receives some garbled circuit GC and the encoded input $\mathcal{L}_z$. $\mathcal{R}$ feeds the each i-th simulator $\mathcal{S}^i_{\mathsf{OT}}$ with $\mathcal{L}^i_z$ and provides $\mathcal{A}$ with GC along with a random w and a key $\mathsf{sk} \leftarrow \mathsf{MACGen}(1^\lambda)$. The adversary replies with a $Y_{z'}$ that the reductions forwards to the challenger.

It is easy to see that the reduction is efficient and that the inputs that $\mathcal{R}$ sends to $\mathcal{A}$ are correctly distributed. By assumption we have that $\mathsf{De}(d_z, Y_{z'}) = z' \neq \perp$ and $z' \neq z$ with probability $\epsilon(\lambda)$. This implies that the reduction succeeds in the authenticity game with the same probability. This is a contradiction and proves that the two experiments are indinstinguishable.

$\mathcal{S}_{|z|+1} \approx \mathcal{S}_{|z|+2}$: Assume towards contradiction that there exists an adversary $\mathcal{A}$ that can distinguish between two simulations with probability $\epsilon(\lambda)$, for some non-negligible function ϵ. Then we can construct the following reduction against the simulation obliviousness of the garbling scheme.

$\underline{\mathcal{R}(1^\lambda)}$: $\mathcal{R}$ simulates the experiment as specified in $\mathcal{S}_{|z|+1}$ until the simulation of the parallel OTs where it uses the extractor to learn the z, then it sends to its challenger the tuple $(\nabla, x, \mathcal{L}_x, z)$. As a response it receives some garbled circuit GC and the encoded input $\mathcal{L}_z$. $\mathcal{R}$ feeds the i-th simulator $\mathcal{S}^i_{\mathsf{OT}}$ with $\mathcal{L}^i_z$ and provides $\mathcal{A}$ with GC along with a random w and a key $\mathsf{sk} \leftarrow \mathsf{MACGen}(1^\lambda)$. The rest of the execution is unchanged.

It is easy to see that the simulation is efficient and that in case of $b = 0$, the garbled circuit GC is sampled as $\mathsf{Gb}(1^\lambda, |z|, e)$ and therefore the simulation is identical to $\mathcal{S}_{|z|+1}$. On the other hand if $b = 1$ then $\mathsf{GC} \leftarrow \mathcal{S}_{\mathsf{ob}}(1^\lambda, |z|, \mathcal{L}_x || \mathcal{L}_z)$, which means that the simulation perfectly reproduces the inputs of $\mathcal{A}$ in $\mathcal{S}_{|z|+2}$. It follows that the reduction correctly guesses the coin b with probability greater than $1/2 + \epsilon(\lambda)$, which is a contradiction.

$\mathcal{S}_{|z|+2} \approx \mathcal{S}_{|z|+3}$: Assume towards contradiction that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathcal{S}_{|z|+2}$ and $\mathcal{S}_{|z|+3}$ with probability $\epsilon(\lambda)$, for some non-negligible function ϵ. Then we can construct the following reduction against the simulation privacy of the garbling scheme.

$\underline{\mathcal{R}(1^\lambda)}$: $\mathcal{R}$ simulates the experiment as specified in $\mathcal{S}_{|z|+2}$ until the simulation of the parallel OTs where it uses the extractor to learn the z, then it sends to its challenger the tuple $(\nabla, x, \mathcal{L}_x, z)$. As a response it receives some garbled circuit GC the encoded input $\mathcal{L}_z$ and the decoding table d. $\mathcal{R}$ feeds the i-th simulator $\mathcal{S}^i_{\mathsf{OT}}$ with $\mathcal{L}^i_z$ and provides $\mathcal{A}$ with GC along with a random w and a key $\mathsf{sk} \leftarrow \mathsf{MACGen}(1^\lambda)$. The rest of the execution is unchanged.

It is easy to see that the simulation is efficient and that in case of $b = 0$, the garbled circuit GC is sampled as $\mathsf{Gb}(1^\lambda, \nabla, e)$ and therefore the simulation is identical to $\mathcal{S}_{|z|+2}$. On the other hand if $b = 1$ then $\mathsf{GC} \leftarrow \mathcal{S}_{\mathsf{prv}}(1^\lambda, |z|, f(x, z), \mathcal{L}_x || \mathcal{L}_z)$, which means that the simulation perfectly reproduces the inputs of $\mathcal{A}$ in $\mathcal{S}_{|z|+3}$. It follows that the reduction correctly guesses the coin b with probability greater than $1/2 + \epsilon(\lambda)$, which is a contradiction.

$\mathcal{S}_{|z|+3} \approx \mathcal{S}_{\mathsf{C}}$: We observe that the two executions are identical except that the decision whether to allow the computation of f on z is outsourced to $\mathcal{F}$. Therefore the two experiments are trivially indistinguishable to the eyes of the adversary.

□

C Modifications on Obliv-C

Let us first recall how Obliv-C works and give a high-level overview of the implementation of METIS as a basis for our modifications presented afterwards. Obliv-C consumes C-like source code specifying the function f which should be evaluated securely. This function is then compiled into a circuit representation and used for executions of Yao's protocol. The specification of the function f is enabled by two special constructs: *obliv-qualified types* – variables of these types are encrypted and can only be revealed if both parties agree – and *oblivious conditional*. With these two primitives we can specify the evaluation functions employing the obliv-qualified types and oblivious conditionals. For instance, the number of variants can be computed by looping over the variant bits of the encoding. Each bit is represented by an obliv-qualified bool. Using the oblivious conditional it is checked whether one of the variant bits is set to true (because 00 means there is no variant). If this is the case, an obliv-qualified counter is incremented.

Furthermore, Obliv-C provides methods allowing the parties to pass their inputs to the oblivious computation and receive the output. This methods behave differently depending on which party executes them. The method *obliv bool feedOblivBool*(*bool input*, *int party*) takes as first argument the input to feed in and the party which is required to distinguish whether the input comes from the garbler or the evaluator as the second input. The return type is an obliv-qualified *bool*.

In a nutshell, *feedOblivBool* converts the input into its obliv-qualified counterpart which is then used in the computations of the function. In METIS these booleans tell whether the corresponding bits of the encoding are set. The (unmodified) method's behavior depends on which party executes it and from which party the input originates. We have to distinguish the following four cases:

- Execution by the garbler & input comes from the garbler: The garbler chooses the labels and sends the label corresponding to its input to the evaluator.
- Execution by the garbler & input comes from the evaluator: The garbler chooses the labels and executes oblivious transfer such that the evaluator only receives the label corresponding to its input.
- Execution by the evaluator & input comes from the garbler: The evaluator receives the label corresponding to the garbler's input.

– Execution by the evaluator & input comes from the evaluator: The evaluator executes oblivious transfer to receive the label corresponding to its input.

Let us highlight the difference between the METIS and the unmodified version of Yao's protocol: In the standard version of Yao's protocol, the garbler is generating all labels. However in our setting, the server M in METIS is provided with the labels corresponding to the data owner's genetic information from the sequencing center S. Additionally these labels are independently computed by the client C.

To account for the different flow of these labels, we added a function *obliv bool feedOblivBoolByLabel*(*char* **label*): the function has only one input, a bitstring instead of a boolean. We do not include the party, as in METIS only the input from the data owner is using this special construction. The garbler is inputting the label corresponding to zero (see the discussion in subsection 5.3 for why this is sufficient) and the evaluator feeds the label it obtained from evaluating the PRF. For the additional inputs (for example, when computing the number of variants in a region, the region boundaries are the additional inputs) we keep the methods provided by Obliv-C.

We implemented METIS instantiated with the four functions (SNP, insertion, number of variants and frameshift) by implementing them in Obliv-C and executing Yao's protocol, while our modification of the *feedOblivBool* method allowed to directly feed the input labels.