Private Computation with Genomic Data for Genome-Wide Association and Linkage Studies

Ali Shahbazi[1], Fattaneh Bayatbabolghani[1], and Marina Blanton[2*]
[1] **Department of Computer Science and Engineering, University of Notre Dame**
[2] **Computer Science and Engineering Department, University at Buffalo, The State University of New York**
[1] **{ashahbaz, fbayatba}@nd.edu,** [2]**mblanton@buffalo.edu**

Abstract

Genome-wide association studies (GWAS) are currently the most powerful tool for assessing associations between common single-nucleotide polymorphisms (SNPs) and common genetic diseases such as auto-immune disease, heart disease, diabetes, and others. Because genomic data is very sensitive due to the possibility of revealing information not only about the data owner but also about his or her relatives, protection of genomic data is highly recommended. In this work, we put forward a secure realization of a suite of statistical tests used in both genome-wide association and linkage studies. The optimizations are tailored toward the secret sharing setting which is suitable for both secure multi-party computation as well as secure computation outsourcing to multiple servers. The goal is to aid adoption of secure computation techniques for the purposes of genome analysis by covering the chain of statistical tests used in both GWAS and GWLS. We implement the developed protocols and demonstrate their efficiency.

1 Introduction

With rapid advances in genome sequencing, studying of genomic data has seen a dramatic increase in the recent years in a variety of applications and fields as diverse as medicine, bio-technology, anthropology, and social sciences. Genome-wide association studies (GWAS) examine many common genomic variations in different individuals to determine if any variant is associated with a trait. GWAS play a crucial role in medicine and the pharmaceutical industry. In particular, in medicine information obtained through GWAS is used to detect, treat, and prevent certain diseases such as asthma, cancer, diabetes, immune disease, heart disease, and others. GWAS enables personalized medicine, which allows one to choose the most effective treatment for an individual based on his or her genetic makeup. GWAS was also recently used for pharmaceutical and other medical purposes such as drug safety [1], drug repositioning [2, 3], identifying rare, metabolic or auto-immune disease, and identifying genetic variants in non-coding regions of the genome that are associated with complex human disease [4, 5]. The increased use of GWAS in drug and health related topics serves as the motivation for this work.

Studying genomic data often requires genomes to be shared. This is because collecting enough samples of individuals carrying a trait can be difficult for a single hospital or clinic, especially when studying a rare disease. Furthermore, due to the size of a human genome, GWAS computation over a large dataset is computation-intensive and thus it may be desirable to delegate storage of genomic data and computation over it to a third party such as a cloud provider.

Releasing genomic data to third parties, however, presents a challenge due to the highly sensitive nature of such data. Even if an individual consents to sharing of his or her genetic data, his or her genome reveals not only information about that individual, but also about the individual's relatives. There are multiple examples when releasing anonymized genomic data was shown to be dangerous. This is because of the ability to re-identify the individuals whose genomic information is stored in a dataset by statistical means. For example, a statistical method for analyzing DNA samples published in [6] led the US National Institutes of Health (NIH) to remove anonymized genomic data from its public databases [7]. Others showed additional attacks that made it easier to recover identifying information from anonymized genomic datasets and learn sensitive information about them (see, e.g., [8, 9, 10, 11, 12, 13, 14]). Thus, alternative solutions need to be pursued.

With this motivation, in this work we focus on privacy-preserving computation used in a genome-wide association studies (GWAS) and genome-wide linkage studies (GWLS), both of which study how genes influence traits. GWAS

[*] Work done while at the University of Notre Dame.

analyzes frequencies of sequences observed in certain locations in a genome in case and control models. In GWLS, on the other hand, a single model is evaluated against different locations in a genome to assess the probability of the disease for each genotype (specific collection of genes). In addition, there is a new area of interest in genome studies that take advantage of both methods: association and linkage studies. For example, GWLS reveals novel loci (locations in a genome) for some complex diseases such autism, prenatal conditions, and others, while GWAS establishes a pattern of these mutations and genome specifications in different populations and different phenotypes (observable characteristics).

To accomplish this goal, we provide a suit of statistical tests run in a sequence on genomic data. This includes testing for Hardy-Weinberg equilibrium (HWE), linkage disequilibrium (LD), Cochran-Armitage test for trend (CATT), and Fisher test. We choose a flexible framework for privately computing with genomic data that allows for both secure joint computation by multiple entities (such as hospitals) as well as secure computation outsourcing to a number of computational servers. We also implement the proposed techniques and show their efficiency through experimental results.

Note that we focus on a powerful set of fundamental tests (HWE, LD, CATT, and Fisher's tests), while there are other approaches used in GWAS such as logistic regression or logistic linear models, or even more complex algorithms that employ neural networks or machine learning. Alternative approaches might be preferred in practice due to their ability to handle covariates, but such tests are beyond the scope of this work. Several recent publications used CATT, LD and Fisher's tests for genome-wide studies in a setting that targets achieving privacy [15, 16, 17, 18]. For example, Ghodsi et.al [15] recently introduced a boosted version of the Cochran-Armitage test, which could be covered by our solution with minor changes. Our constructions offer simplicity and efficiency, which might be preferred in certain environments.

2 Related Work

During recent years, different aspects of genomic privacy have been studied in a variety of settings. The first type of publications focuses on recovering private information by analyzing public genomic and related data. In a seminal study [6], Homer et al. found that by simply using statistical methods the presence of an individual in a given genetic dataset can be determined. Gymrek et al. [13] showed how the identity of a genome data owner can be recovered from anonymized genetic data. Erlich and Narayanan [19] consequently categorized different genetic privacy breaching techniques and discussed their complexity and maturity. Most recently, Ruichu et al. [20] described a practical privacy attack that can identify individuals from a specific dataset and proposed a solution to prevent this attack.

Another type of publications have protection of genomic data prior to its release as their goal. In particular, there are approaches that utilize access control mechanisms in a database or data anonymization techniques such as [21, 22, 23, 24]. For example, Agrawal et al. [21, 22] utilize the so-called trust-but-verify approach, in which users can execute certain types of queries based on their privileges, but are not permitted to download the data. k-anonymity is a popular data anonymization technique applied to sensitive data prior to its release to lower the possibility of re-identification of the subjects contained in the database [23, 24]. The technique has been used to anonymize medical data before disclosure, but there are challenges with applying it to DNA data of high dimensionality [25].

The last category of publications (e.g., [26, 27, 28, 29, 30, 31] among others) deal with protecting privacy of genomic data via secure computation mechanisms when it is being used in computation; this is also what we pursue in this work. They use different types of computation such as pattern matching, sequence alignment, and statistical tests used in GWAS, which we briefly describe next.

Work on secure pattern matching with applications to genomic computation include [29, 28, 32, 33]. For instance, Katz et al. [28] apply secure text processing techniques to DNA matching. De Cristofaro et al. [32] also provide an efficient private pattern matching protocol that additionally hides the size and position of the markers being searched for in a DNA sequence.

Publications that focus on secure evaluation of the edit distance, Hamming distance, and sequence alignment include [26, 30, 27, 31, 34, 35, 36, 37, 38]. As an example, Atallah et al. published a series of investigations on secure computation outsourcing mostly for the edit distance [34, 35, 36, 37]. Blanton et al. [38] presented a highly efficient approach for sequence comparison outsourcing to two servers.

Work on secure evaluation of GWAS statistical tests can be divided into two categories: secure outsourcing to a single server using homomorphic encryption and secure computation using two or more servers (for the purposes of secure multi-party computation or secure outsourcing). From the first category, [39, 40, 41] treat a number of different GWAS tests evaluated on encrypted genomic data, while [42] focuses on predictive analysis for medical applications. In addition, as part of the 2015 iDASH secure genome analysis competition, publications [43, 44, 45] perform secure outsourced computation of the chi-squared test and minor allele frequencies, as well as the Hamming and edit distances adopted to genome sequences in [44], using homomorphic encryption. Publications from the second category include [46, 47, 48]. The first two of these publications report on their experience in devising secure computation solution for the 2015 iDASH competition using secret sharing and garbled circuit evaluation, respectively. The last publication, Kamm et al. [48], is the closest to our work and treats a number of statistical tests used for GWAS using secret sharing techniques. In particular, [48] considered two χ^2 tests for independence, CATT, and the transmission disequilibrium test (TDT), which is applicable only to parent-child trios. The only test that overlaps with our work is CATT, as the goals of our work and [48] somewhat differ. The goal of our work is to provide a comprehensive suite of secure statistical tests that could be used with genome-wide association or linkage studies. The HWE and LD tests are necessary components of such data analysis that ensure quality of the data prior to analyzing via CATT or Fisher tests. Neither these components nor the Fisher test commonly used with GWAS/GWLS were covered by [48]. On the other hand, that work treats certain aspects of running secure genome data analysis not covered in this work. We comment on the performance of our tests and those from [48] in section 7.

3 Genomic Background

This section describes the (non-secured) computation which is the focus of this work, namely, statistical tests for association and linkage studies. Before we proceed with the details of the tests, we briefly describe genomic terms and notation used throughout this work.

3.1 Definitions and Notation

A *DNA* is a sequence of nucleotides $\{A, C, G, T\}$. An individual's collection of genes is called a *genotype* and physical observable characteristics of an individual are called a *phenotype*. A *genetic marker* is defined as a gene or a DNA segment with a known locus (location) on a chromosome, which is typically used to help link an inherited disease with the responsible gene. Then a set of closely linked genetic markers found in one chromosome that tend to be inherited together is called a *haplotype*.

A *single nucleotide polymorphism* (SNP) represents a common type of a genetic variation among people in a single nucleotide that occurs at a specific locus in a genome. One of a number of alternative forms of a gene at a given locus is called an *allele*. The most and the least common alleles that occur in a given population are called *major* and *minor* alleles, respectively. We denote a major allele by a capital letter, e.g., A, and a minor allele by the corresponding lowercase letter, e.g., a. An individual inherits two alleles for each gene, one from each parent. If the two alleles are the same, the individual is *homozygous* for that gene and is *heterozygous* otherwise. Based on that information, we distinguish between the following categories: homozygous reference genotype, denoted as AA; heterozygous genotype, denoted as Aa; and homozygous variant genotype, denoted as aa. We refer to the two alleles inherited for a particular gene as a genotype.

Let N denote the total number of collected alleles in a pool of genes. We then use N_A and N_a to denote the number of major and minor alleles in the observed population, respectively. Similarly, N_{AA}, N_{Aa}, and N_{aa} denote the number of gene variants of the type AA, Aa, and aa, respectively. They are used to compute values N_A and N_a as

$$N_A = 2N_{AA} + N_{Aa}, \qquad N_a = 2N_{aa} + N_{Aa}.$$

In addition, an allele frequency is defined as the number of this allele in a certain locus in the observed population. In other words, we define major and minor allele frequencies p_A and p_a as $p_A = N_A/N$ and $p_a = N_a/N$, respectively. A genotype frequency can be defined analogously.

3.2 Statistical Tests

We next proceed with describing popular tests used in GWAS and GWLS as well as the overall procedure for conducting an association or linkage study.

3.2.1 Hardy-Weinberg Equilibrium

Hardy-Weinberg Equilibrium (HWE) is an equation proposed by G. Hardy and W. Weinberg that states that allele and genotype frequencies in a population will remain constant from generation to generation in the absence of evolutionary influences (such as mutation, selection, etc.). Because such influences are normally present in real populations, the HWE describes an ideal condition against which the effects of these influences can be analyzed. In the case of a single locus with two alleles A and a and corresponding frequencies p_A and p_a, the HWE is stated as:

$$p_A^2 + 2p_Ap_a + p_a^2 = 1.$$

When genotype frequencies of all three genotypes AA, Aa, and aa are known, they can be tested for statistically significant deviations from the HWE. Testing for deviations from the HWE is typically performed using (Pearson's) chi-squared test, χ^2, using the observed genotype frequencies and the expected frequencies obtained from the Hardy-Weinberg principle. Fisher exact test, however, can be used instead when χ^2 has small expected values and the asymptotic assumption of the chi-squared distribution no longer holds (this can be the case in systems with a large number of alleles and an insufficient number of individuals to represent all genotype classes).

To use the chi-squared test for determining a deviation from the HWE, we compute:

$$\chi^2 = \sum_{i \in \{AA, Aa, aa\}} \frac{(N_i - E_i)^2}{E_i}, \tag{1}$$

where E_i's represent expected values of the genotypes, defined as $E_{AA} = (N_A)^2/(4N)$, $E_{Aa} = (N_AN_a)/(2N)$, and $E_{aa} = (N_a)^2/(4N)$. If the computed χ^2 is larger than a particular threshold, the data is considered to be unfit and is rejected. This means that different input data needs to be collected or a different locus should be chosen for the study.

3.2.2 Linkage Disequilibrium

Linkage Disequilibrium (LD) is an important notion in population genetics and it occurs when genotypes at two different loci are not independent of each other. In other words, LD is the non-random association of pairs of alleles that often descend from a single ancestral chromosome. Consider two loci A and B with two alleles each (A, a, B, and b). There are 9 possible genotypes $AABB$, $AABb$, $AAbb$, $AaBB$, $AaBb$, $Aabb$, $aaBB$, $aaBb$, $aabb$, and there are four haplotypes AB, Ab, aB, ab. Let us use N_{AB}, N_{Ab}, N_{aB}, and N_{ab} as the number of instances of each of the four haplotypes in the observed population. Then, their population frequencies are computed as:

$$p_{AB} = \frac{N_{AB}}{N}, \qquad p_{Ab} = \frac{N_{Ab}}{N}, \qquad p_{aB} = \frac{N_{aB}}{N}, \qquad p_{ab} = \frac{N_{ab}}{N}.$$

When the alleles' frequencies are independent (i.e., we have linkage equilibrium), we expect that:

$$p_{AB} = p_Ap_B, \qquad p_{Ab} = p_Ap_b, \qquad p_{aB} = p_ap_B, \qquad p_{ab} = p_ap_b.$$

where, as before, $p_A = N_A/N$, $p_a = N_a/N$ and similarly $p_B = N_B/N$, $p_b = N_b/N$, but now $N_A = N_{AB} + N_{Ab}$, $N_a = N_{aB} + N_{ab}$, $N_B = N_{AB} + N_{aB}$, $N_b = N_{Ab} + N_{ab}$. However, if the alleles are in LD, the formulas become:

$$p_{AB} = p_Ap_B + D_{AB}, \qquad p_{Ab} = p_Ap_b - D_{AB}, \qquad p_{aB} = p_ap_B - D_{AB}, \qquad p_{ab} = p_ap_b + D_{AB}.$$

The parameter D_{AB} is called the coefficient of LD and can be computed as $D_{AB} = p_{AB} - p_A p_B$. Chi-square statistics for the hypothesis H_0 of no disequilibrium (i.e., $D_{AB} = 0$) is computed as:

$$\chi^2_{A,B} = \frac{2N(D_{AB})^2}{p_A(1-p_A)p_B(1-p_B)} = \frac{2N(D_{AB})^2}{p_A p_a p_B p_b}. \tag{2}$$

H_0 is rejected (i.e., LD is present) if $\chi^2_{A,B}$ exceeds a particular threshold.

3.2.3 Cochran-Armitage Test for Trend

Statistical tests for trends, including Cochran-Armitage test for trend (CATT), represent a modification of (Pearson's) chi-squared test and are used to assess the presence of association between a variable with two different categories (cases and controls) and a variable with m different categories. In application to GWAS, we have three different categories for the second variable, which are AA, AB, and BB if association between two alleles is being tested or AA, Aa, and aa if the study is for one allele. The table below provides an example of information used in a CATT test, where in each N_{ij} i indicates the row and j the column.

	Group 0	Group 1	Group 2	Total
Controls	N_{00}	N_{01}	N_{02}	R_0
Cases	N_{10}	N_{11}	N_{12}	R_1
Total	C_0	C_1	C_2	N

Using two categories for the first variable and three categories for the second variable, we can represent the Cochran-Armitage formula as:

$$T = \sum_{i=0}^{2} w_i(N_{0i}R_1 - N_{1i}R_0),$$

where $w = (w_0, w_1, w_2)$ corresponds to predetermined weights. There are three standard values of w: $(0,1,2)$ is used for the co-dominant model, $(0,1,1)$ for the dominant model, and $(0,0,1)$ for the recessive model.

To perform CATT, one computes the chi-squared test as:

$$\chi^2 = \frac{T^2}{Var(T)}, \tag{3}$$

where the variance of T is given by the formula:

$$Var(T) = \frac{R_0R_1}{N}\left(\sum_{i=0}^{2} w_i^2 C_i(N-C_i) - 2\sum_{i=0}^{1}\sum_{j=i+1}^{2} w_i w_j C_i C_j\right).$$

Once χ^2 is computed, it is compared to a particular threshold to determine whether the trend in question is present.

3.2.4 Fisher Test

Fisher's exact test is another statistical significance test used in the analysis of contingency tables similar to CATT. The Fisher test is more accurate than chi-squared tests when sample sizes are small and is preferred in those circumstances (e.g., when a genomic test is to be run on members of a single family). Unlike chi-squared tests that approximate the result, the Fisher test computes the exact values. The Fisher test is valid for all sample sizes, but the computation becomes prohibitively expensive when sample sizes are very large.

In the Fisher test, the contingency table for assessing the presence of association between a variable with k different categories and a variable with m different categories looks like the following:

	Group 1	Group 2		Group m	Total
Category 1	N_{11}	N_{12}	$\dots$	N_{1m}	R_1
Category 2	N_{21}	N_{22}	$\dots$	N_{2m}	R_2
Category k	N_{k1}	N_{k2}	$\dots$	N_{km}	R_k
Total	C_1	C_2	$\dots$	C_m	N

The p-value of the Fisher test is computed by the formula:

$$p = \frac{\left(\prod_{i=1}^{k} (R_i!)\right) \cdot \left(\prod_{i=1}^{m} (C_i!)\right)}{N! \cdot \prod_{i=1,j=1}^{k,m} (N_{ij}!)}.$$

As before, the computed value needs to be compared to a predefined threshold to determine whether the trend is present.

In the case of genome-wide studies, the Fisher test is often used for both GWAS and pharmaceutical drug tests. Then to perform the test over a table with two categories of cases and controls and two groups of A and B alleles, we can rewrite the table as:

	A	B	Total
Controls	N_{0A}	N_{0B}	R_0
Cases	N_{1A}	N_{1B}	R_1
Total	C_A	C_B	N

Here controls correspond to category 0 and cases to category 1. Now computation of the p-value simplifies to:

$$p = \frac{R_0! \cdot R_1! \cdot C_A! \cdot C_B!}{N! \cdot N_{0A}! \cdot N_{0B}! \cdot N_{1A}! \cdot N_{1B}!}. \quad (4)$$

3.2.5 Overall GWAS and GWLS Procedures

Having specified the necessary statistical tests, we conclude this section with the description of the overall procedure used in GWAS or GWLS.

The HWE and LD tests are used to assess the quality of input data sets. HWE is invoked with a single allele, while LD is invoked for a pair of alleles. If the appropriate quality tests passes, statistical significance of the hypothesis is assessed using the CATT or Fisher test. With a single allele, the datasets are normally large enough so that the CATT test is used (while the Fisher test is applicable as well). With a pair of alleles, however, the dataset may become partitioned into many categories so that individual sample sizes are small, in which case the Fisher test would be preferred.

To summarize, with a single allele, we typically first invoke the HWE test and reject the data if the test fails. Otherwise, CATT is run on the gathered data. With a pair alleles, we invoke the LD test and reject the data if the test does not pass. Otherwise, the Fisher test is executed (or CATT can be used as well if sample sizes are sufficiently large).

4 Security Model

Often genomic data is collected and resides at medical centers or other facilities which are responsible for keeping this information confidential. Because it is often desirable to aggregate the data from multiple medical centers to provide stronger statistical results, the medical centers will benefit from participating in privacy-preserving computation, which will evaluate the necessary statistical functions without sharing any unintended information with other participants. In such a setting, we deal with multiple data owners who engage in secure multi-party computation or delegate the secure computation to a number of computational servers who run the computation on their behalf without learning any information that they handle in the process of the computation. Thus, we frame secure computation in a general setting where there are a number of input providers, a number of computational parties, and a number of output recipients. The input providers privately enter their data in the computation, the computational parties evaluate the necessary function(s) on the private data, and the computed outputs are revealed to the output recipients (possibly different outputs to different parties). These three sets of participants can be formed in an arbitrary way. This, for instance, would allow a number of medical centers directly engage in secure computation, or choose a subset of them to carry the computation on the combined data on behalf of the group, or securely outsource the computation to a number of independent servers. This setup can also handle the setting when a single data owner employs a number of computational servers for executing the tests for reasons of limited resources or expertise.

Security of any multi-party protocol (with two or more participants) can be formally shown according to one of the two standard security definitions. The first, weaker security model assumes that the participants are semi-honest (also known as honest-but-curious or passive), defined as they follow the computation as prescribed, but might attempt to learn additional information about the data from the intermediate results. The second, stronger security model deals with malicious (also known as active) participants who are allowed to arbitrarily deviate from the prescribed computation. The focus of this work is on the semi-honest model. The techniques that we employ, however, can be extended to support the stronger malicious model as well using well-known results. Security of a protocol in our setting is then defined as follows.

Definition 1 *Let parties $P_1, \ldots, P_n$ with pair-wise secure channels engage in a protocol π that computes a (possibly probabilistic) n-ary function $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$, where P_i contributes input in_i and receives output out_i. Let $\mathrm{VIEW}_\pi(P_i)$ denote the view of participant P_i during the execution of protocol π. More precisely, P_i's view is formed by its input and internal random coin tosses r_i, as well as messages $m_1, \ldots, m_k$ passed between the parties during protocol execution: $\mathrm{VIEW}_\pi(P_i) = (\mathsf{in}_i, r_i, m_1, \ldots, m_k)$. Let $I = \{P_{i_1}, P_{i_2}, \ldots, P_{i_t}\}$ denote a subset of the participants, $\mathrm{VIEW}_\pi(I)$ denote the combined view of participants in I during the execution of protocol π (i.e., $\mathrm{VIEW}_\pi(I) = (\mathrm{VIEW}_\pi(P_{i_1}), \ldots, \mathrm{VIEW}_\pi(P_{i_t}))$), and $f_I(\mathsf{in}_1, \ldots, \mathsf{in}_n)$ denote the projection of $f(\mathsf{in}_1, \ldots, \mathsf{in}_n)$ on the coordinates in I (i.e., $f_I(\mathsf{in}_1, \ldots, \mathsf{in}_n)$ consists of the i_1th, ..., i_tth elements that $f(\mathsf{in}_1, \ldots, \mathsf{in}_n)$ outputs). We say that protocol π is t-private in the presence of static semi-honest adversaries if for each coalition I of size at most t and all $\mathsf{in}_i \in \{0,1\}^*$ there exists a probabilistic polynomial time simulator S_I such that $\{(S_I(\mathsf{in}_I, f_I(\mathsf{in}_1, \ldots, \mathsf{in}_n)), f(\mathsf{in}_1, \ldots, \mathsf{in}_n))\} \equiv \{(\mathrm{VIEW}_\pi(I), (\mathsf{out}_1, \ldots, \mathsf{out}_n))\}$, where $\mathsf{in}_I = (\mathsf{in}_{i_1}, \ldots, \mathsf{in}_{i_t})$ and "$\equiv$" denotes computational or statistical indistinguishability.*

Note that this definition restricts the information that the computational parties learn by participating in secure computation. Thus, if participant P_i contributes no input and/or learns no output, its respective in_i and/or out_i will be empty. Also note that in the semi-honest setting the participants supply their inputs truthfully.

5 Building Blocks

Secure multi-party computation and outsourcing can be realized using a number of different underlying techniques such as homomorphic encryption, garbled circuit evaluation, or secret sharing. In this work we choose to utilize linear threshold secret sharing techniques due to the flexibility in the settings they support. In particular, such techniques naturally support any number of input providers and output recipients with n computational parties, where typically $n \geq 3$. Then before the computation takes place each data owner secret-shares each of her data item among the computational parties and upon computation termination output shares are communicated to each output party who locally reconstructs the result.

Shamir secret sharing [49] is a popular (n,t)-threshold linear secret sharing scheme. If a secret value is secret-shared among n parties, any t or fewer of them can learn no information about the secret in the information-theoretic sense, while combining $t+1$ or more shares allows for the result to be reconstructed. As with any linear secret sharing scheme, any linear combination of secret shared values can be computed locally by each party. Multiplication of secret-shared values requires communication and is treated as an elementary interactive operation. To support multiplications, it is requires that $n \geq 3$ and $t < n/2$. Secret shares are represented as integers in a finite field $\mathbb{F}_p$ (normally with prime p)

Protocol	Rounds	Interactive operations
Add/Sub	0	0
Mul	1	1
LT	4	$4\ell - 2$

Table 1: Complexity of building blocks using secret sharing.

of sufficient size to represent all values. No expensive cryptographic operations are used and this is why performance is measured in the number of interactive operations and also in the number of sequential operations or rounds.

The basic techniques are t-private in the presence of semi-honest participants. There are, however, a variety of general techniques that extend the basic solution to guarantee security in the presence of malicious participants.

In what follows, we list operations which we use as building blocks in constructing secure protocols for GWAS and GWLS. These sub-protocols are available using a number of different underlying techniques, but we list their complexities using Shamir secret sharing. In what follows, notation $[x]$ denotes that the value of x is protected (via secret sharing) and is not available to the participants in the clear. All operations are performed on integer values.

- *Addition* $[c] \leftarrow [a] + [b]$ and *subtraction* $[c] \leftarrow [a] - [b]$ are elementary non-interactive operations, which are considered free.
- *Multiplication* $[c] \leftarrow [a] \cdot [b]$ costs 1 interactive operation, while multiplication $[c] \leftarrow a \cdot [b]$ is local (free).
- *Comparison* $[c] \leftarrow \mathsf{LT}([a], [b])$ tests whether $a < b$ and outputs a bit. For ℓ-bit operands a and b, it costs $4\ell - 2$ interactive operations in 4 rounds [50]. The first round is input-independent and can be pre-computed or run simultaneously with the preceding computation. The "less than or equal" operation LTE and comparisons with one private and one public comparands have the same cost.

Complexities of these operations are summarized in Table 1.

6 Secure Computation of Genome-Wide Association and Linkage Studies

In this section, we restructure the computation used in the tests described in section 3 to result in more efficient secure implementations and show how these tests can be securely realized in our secure multi-party computation framework.

Without loss of generality, let the input come from two data owners. It is straightforward to generalize the solution to any number of input providers. Each data owner possesses a collection of DNAs sequences and extracts relevant information from them for the purpose of GWAS or GWLS. We assume that the number of subjects in each collection is known and thus the total number of subjects is public as well. Any data extracted from the DNA sequences, however, remains private and is processed in a protected form by the secure computation protocols. In what follows, we will use notation $x^{(i)}$ to denote the value of variable x held by party i. For example, we have that $N^{(1)} + N^{(2)} = N$.

6.1 Secure Hardy-Weinberg Equilibrium Computation

To perform this test, each input party i contributes $N^{(i)}$, $[N_{AA}^{(i)}]$, $[N_{Aa}^{(i)}]$, $[N_{aa}^{(i)}]$ and the parties also agree on the threshold τ. The computation outputs a bit (fit/unfit) to each input party.

To determine how secure computation is to proceed, let us expand the HWE formula in equation 1 as follows:

$$\chi^2 = \frac{1}{4N} \left(\frac{(4N \cdot N_{AA} - N_A^2)^2}{N_A^2} + \frac{2(2N \cdot N_{Aa} - N_A \cdot N_a)^2}{N_A \cdot N_a} + \frac{(4N \cdot N_{aa} - N_a^2)^2}{N_a^2} \right) \quad (5)$$

In this formula N is public, while the values of all other variables are private. We next note that the quantity in equation 5 is being compared to the threshold τ. Because the division operation is significantly more expensive that integer multiplication in our framework, we can re-write the formula to replace divisions with multiplications. We obtain the following, where all protected variables are marked as such:

$$\left(4N \cdot [N_{AA}] - [N_A]^2\right)^2 [N_a]^2 + 2(2N \cdot [N_{Aa}] - [N_A] \cdot [N_a])^2 [N_A] \cdot [N_a] + (4N \cdot [N_{aa}] - [N_a]^2)^2 [N_A]^2 \leq 4N \cdot \tau \cdot [N_A]^2 \cdot [N_a]^2 \quad (6)$$

Then when the computation commences, the input variables N, $[N_{AA}]$, $[N_{Aa}]$, and $[N_{aa}]$ are computed as the sum of each party's respective values, followed by the computation of $[N_A]$ and $[N_a]$ as specified in section 3.1. Note that all of these operations are local (i.e., considered free). The computation consequently evaluates the expression of equation 6 that involves 10 (interactive) multiplications in 3 rounds, followed by a single comparison. This can be accomplished in $4\ell + 8$ interactive operations in 6 rounds, where ℓ is the bitlength of the values being compared in equation 6 which is proportional to $\log(N)$. At the end of the computation, each computation party communicates its share of the output to each input party (who consequently reconstructs and learns the result).

Because the HWE and other tests are typically evaluated at a number of different locations (alleles), the above described computation will be run in parallel for each location. That is, to perform M HWE tests, the total number of interactive operations increases by a factor of M, but the round complexity remains unchanged.

6.2 Secure Linkage Disequilibrium Computation

To execute this test, each input party i contributes $N^{(i)}$, $[N^{(i)}_{AB}]$, $[N^{(i)}_{Ab}]$, $[N^{(i)}_{ab}]$, and $[N^{(i)}_{aB}]$ and the participants agree on the threshold τ. As with the HWE test, the output of the LD test is a bit communicated to each input party.

Similar to the HWE computation, we expand the LD formula in equation 2 as follows:

$$\chi^2_{A,B} = \frac{2N \cdot D^2}{p_A \cdot p_a \cdot p_B \cdot p_b} = \frac{2N \cdot (N \cdot N_{AB} - N_A \cdot N_B)^2}{N_A \cdot N_a \cdot N_B \cdot N_b} \quad (7)$$

As before, the value of N is public, while all other variables are private, and the resulting value of $\chi^2_{A,B}$ is compared to τ to determine the output. As with the HWE test, we re-structure the computation to avoid the division operation and obtain:

$$2N \cdot (N \cdot [N_{AB}] - [N_A] \cdot [N_B])^2 \leq \tau \cdot [N_A] \cdot [N_a] \cdot [N_B] \cdot [N_b], \quad (8)$$

where $[N_A]$, $[N_B]$, $[N_a]$, and $[N_b]$ are (locally) computed from the input as specified in section 3.2.2 (and for each input variable v its value is locally determined by each computational party as $v = \sum_{i=1}^{2} v^{(i)}$. Thus, the computation involves 4 multiplications in 2 rounds followed by a single comparison as specified in equation 8, which results in $4\ell + 2$ interactive operations in 5 rounds. Also, when the test is executed for M pairs of alleles, the overall work increases by a factor of M, but the round complexity remains unchanged.

6.3 Secure Cochran-Armitage Test for Trend Computation

In the CATT test, the values of N and R_i are public, while each N_{ij} and C_j is private. We also consider only three possible values for the weights $\vec{w} = (w_0, w_1, w_2)$ as described in section 3.2.3 (namely, the values (0, 1, 2), (0, 1, 1), and (0, 0, 1)). Because the weights determine the model used in the computation, we consider two possibilities: (i) the weights are public, as there is only one meaningful model for the tests performed by the input party or parties, and (ii) the weights are private to provide additional protection for the type of tests being performed. In either case, because $w_0 = 0$ for any model considered in this work, we expand the CATT computation in equation 3 as:

$$\chi^2 = \frac{T^2}{Var(T)} = \frac{N(w_1 \cdot (N_{01} \cdot R_1 - N_{11} \cdot R_0) + w_2 \cdot (N_{02} \cdot R_1 - N_{12} \cdot R_0))^2}{R_0 R_1 (w_1^2 \cdot C_1 \cdot (N - C_1) + w_2^2 \cdot C_2 \cdot (N - C_2) - 2w_1 \cdot w_2 \cdot C_1 \cdot C_2)} \quad (9)$$

For this test, the threshold τ is publicly known and each input party i contributes $N^{(i)}$, $R^{(i)}_0$, $R^{(i)}_1$ $[N^{(i)}_{01}]$, $[N^{(i)}_{11}]$, $[N^{(i)}_{02}]$, and $[N^{(i)}_{22}]$. In the case of public weights, w_1 and w_2 are provided as public constants, and in the case of private weights, one input party supplies $[w_1]$ and $[w_2]$.

When the computation commences, N, R_0, R_1, $[N_{01}]$, $[N_{11}]$, $[N_{02}]$, and $[N_{22}]$ are computed as the sum of each input party's respective values, followed by the computation of $[C_1]$ and $[C_2]$ as specified in section 3.2.3. All of this computation is local. Consequently, the computational parties evaluate the expression

$$N \cdot ([w_1] \cdot ([N_{01}] \cdot R_1 - [N_{11}] \cdot R_0) + [w_2] \cdot ([N_{02}] \cdot R_1 - [N_{12}] \cdot R_0))^2 \leq \\ R_0 R_1 \tau \cdot ([w_1]^2 \cdot [C_1] \cdot (N - [C_1]) + [w_2]^2 \cdot [C_2] \cdot (N - [C_2]) - 2[w_1] \cdot [w_2] \cdot [C_1] \cdot [C_2]) \quad (10)$$

where the weights are assumed to be private. This computation involves 8 multiplications in 2 rounds followed by a single comparison and the total cost is $4\ell + 6$ interactive operations in 5 rounds.

When the weights w_1 and w_2 are public and non-zero, evaluation of equation 10 simplifies to 4 (interactive) multiplications in 1 round followed by a comparison, and the total cost becomes $4\ell + 2$ interactive operations in 4 rounds. Furthermore, when $w_1 = 0$ (i.e., the recessive model with $\vec{w} = (0, 0, 1)$, the cost reduces to 4ℓ interactive operations.

As before, we can evaluate M CATT tests at different locations in parallel without increasing the round complexity.

6.4 Secure Fisher Test Computation

In the Fisher test, as before, the values of N, R_0, R_1, and τ are public and the remaining inputs are private. To make the computation suitable for efficient secure evaluation, we proceed with computing the logarithm of the p-value instead of directly implementing the operations in equation 4. This allows us to avoid working with values of excessive bitlength and also allows us to replace the division operation with a very fast subtraction operation. Thus, the computation becomes:

$$\log(p) = \log(R_0!) + \log(R_1!) + \log(C_A!) + \log(C_B!) - \log(N!) \\ - \log(N_{0A}!) - \log(N_{0B}!) - \log(N_{1A}!) - \log(N_{1B}!) \quad (11)$$

The value of $\log(R_0!) + \log(R_1!) - \log(N!)$ is publicly known and is simply added to the expression. For each private value v, we compute the logarithm of $v!$ as the sum of logarithms from 1 until the value of v. Because each v is private, we cannot reveal the number of terms in the sum and instead need to iterate until the upper bound of v is reached, i.e., $\sum_{i=1}^{v_{max}} \log(i)$. Now, because the upper bound v_{max} is typically larger than the value that v takes, we need to compare the current value of i to v and include $\log(i)$ in the sum only if $i \leq v$. This oblivious computation of $\log(v!)$ for some private v is given in the code below:

```
$[res] = 0$;
for $i = 2, \ldots, v_{max}$
    $[c] = \mathsf{LTE}(i, [v])$;
    $[res] = [res] + [c] \cdot \log(i)$;
```

This computation implements the logic "if $(i \leq [v])$ then $[res] = [res] + \log(i)$." We assume that $v > 0$ for each private v used in the computation, as otherwise the tests are not meaningful and $\log(0) = -\infty$. There is also no need to compare v to 1 because $\log(1) = 0$ and does not affect the result. For different private variables, the upper bounds will differ. For example, both N_{0A} and N_{0B} are upper-bounded by R_0, N_{1A} and N_{1B} are upper-bounded by R_1, and C_A and C_B are upper-bounded by N.

To further optimize the computation, we next notice that we can simultaneously compute $\log([N_{0A}]!)$ and $\log([N_{0B}]!)$ using one set (as opposed to two sets) of R_0 comparisons, which are the most expensive operations in the above computation. In detail, the code to be executed now becomes:

```
$[v_A] = [v_B] = 0$;
for $i = 2, \ldots, R_0 - 1$
    $[c_i] = \mathsf{LTE}(i, [v])$;
    $[v_A] = [v_A] + [c_i] \cdot \log(i)$;
    $[v_B] = [v_B] + (1 - [c_i]) \cdot \log(R_0 + 1 - i)$;
```

where v_j denotes the value of computed $\log(N_{0j}!)$ for $j = A, B$. In other words, the logic is "if $(i \leq [N_{0A}])$ then $[v_A] = [v_A] + \log(i)$ else $[v_B] = [v_B] + \log(i)$." Note that this correctly computes the values because $N_{0A} + N_{0B} = R_0$ and the value of N_{0B} is used only implicitly. As before, we assume that N_{0A} and N_{0B} are non-zero. We use notation c_i to highlight the fact that all comparisons are independent of each other and can be performed simultaneously, while (local) addition at the end is performed sequentially.

The same logic is used for computing $\log(N_{1A}!)$ and $\log(N_{1B}!)$ (resp., $\log(C_A!)$ and $\log(C_B!)$) using R_1 (resp., N) in place of R_0 and N_{1A} (resp., C_A) in place of N_{0A}.

To summarize the Fisher test computation, each input party i contributes $[N_{0A}^{(i)}]$ and $[N_{1A}^{(i)}]$ and the parties also input public N, R_0, R_1, and τ. The computation first proceeds with computing $[N_{0A}] = [N_{0A}^{(1)}] + [N_{0A}^{(2)}]$, $[N_{1A}] = [N_{1A}^{(1)}] + [N_{1A}^{(2)}]$, and $[C_A] = [N_{0A}] + [N_{1A}]$. Then the values $\log([N_{0A}]!)$, $\log([N_{0B}]!)$, $\log([N_{1A}]!)$, $\log([N_{1B}]!)$, $\log([C_A]!)$, and $\log([C_B]!)$ are computed as described above using three loops of parallelized operations, where all loops are also executed in parallel. Consequently, $\log([p])$ is computed as in equation 11 and compared to $\log(\tau)$, and the result of the comparison is communicated to the input parties. The overall (non-local) cost is thus near $2N$ simultaneous comparison operations followed by another comparison at the end.

Note that our implementation of securely evaluating $\log(v!)$ proceeds similar to the computation of a table lookup with a private index. Conceptually, the difference is that we (implicitly) store individual values of $\log(i)$ in each position of the array and add all values for i between 1 and v, while with a table lookup the ith element of the array would store $\sum_{j=1}^{i} \log(j)$. Without resorting to advanced techniques of high complexity such as oblivious RAM [51], table lookups with a private index can be implemented by either comparing the private index to all positions of the array or by bit-decomposing the index and using a multiplexer to retrieve the desired element. Both approaches have $O(m \log m)$ complexity and $O(\log m)$ round complexity for an m-element array, which is the same as in our solution. Where our solution becomes beneficial is during the computation of two logarithms at the same time, with majority of the computation being re-used and therefore saved. That is, we compute the values of $\log([N_{0A}]!)$ and $\log([N_{0B}]!)$ using only slightly more work than the computation of $\log([N_A]!)$ alone. The same applies to the computation of $\log([N_{1A}]!)$ and $\log([N_{1B}]!)$ and to the computation of $\log([C_A]!)$ and $\log([C_B]!)$. This optimization does not apply to table lookup-based implementations.

6.5 Security Analysis

Correctness of each protocol is straightforward to verify because each of them follow the original computation. The introduced modifications (such as restructuring the equations to eliminate the division operation or making the computation of the logarithm of a factorial oblivious) do not affect correctness of the computation.

With respect to security, we first note that all operations used in the protocols (namely, integer addition, subtraction, multiplication, and comparisons) have secure implementations as described in section 5. Second, no information that depends on private values is revealed at intermediate points of the protocols. This means that by Canetti's composition theorem [52], composition of secure sub-protocols results in security of the overall constructions.

In more detail, if we assume an implementation based on a (n, t)-threshold linear secret sharing, our protocols inherit the same security guarantees as those of the building blocks (i.e., perfect or statistical security in the presence of secure channels between the parties with at most t corrupt computational parties) because no information about private values is revealed throughout the computation. More formally, to comply with the security definition, it is rather straightforward to build a simulator for our protocols by invoking simulators of the corresponding building blocks to result in the environment that will be indistinguishable from the real protocol execution by the participants. The only protocol that warrants additional discussion is secure implementation of the Fisher test. As discussed in section 6.4, the most non-trivial portion of the computation is oblivious evaluation of logarithms of a factorial. However, if we refer to the computation given in section 6.4, we see that all executed instructions depend only on public values (i.e., the instructions inside each loop are fixed and the number of loop iterations is based on a public variable) and all operations that use private values are realized using secure building blocks. Thus, we obtain that we can build a simulator to show security according to definition 1 for this protocol as well by invoking individual simulators for the underlying secure integer operations.

Although the focus of this work is on semi-honest adversaries, security of our protocols can also be extended to provide provable security guarantees in the presence of malicious participants. In that case, we need to additionally ensure that (i) all computational parties prove that each step of their computation was performed correctly and that (ii) if some dishonest participants quit, others will be able to reconstruct their shares and proceed with the rest of the computation. The above is normally achieved using a verifiable secret sharing scheme (VSS), and a large number of results have been developed over the years (e.g., [53, 54, 55, 56, 57] and others). In particular, because any linear combination of shares is computed locally, each participant is required to prove that it performed each multiplication correctly on its shares. Additional proofs associated with this setting include proofs that shares of a private value were distributed correctly among the participants (when the dealer is dishonest) and proofs of proper reconstruction of a value from its shares (when not already implied by other techniques). In addition, if at any point of the computation the participants are required to input values in a specific form, they would have to prove that the values they supplied are well formed. From the building blocks described in section 5, only comparisons LT/LTE require the computational parties to generate random values of a desired bitlength. Thus, one can use a range proof such as in [58] (secure in the computational setting) for this purpose or create random values of a specified length using an alternative mechanism such as the one suggested in [59] (secure in the information-theoretic sense). These VSS techniques would also work with malicious input parties, who would need to prove that they generate legitimate shares of their data. In general, any technique that can securely and efficiently realize the building blocks we require as well as their secure composition will suffice for the purposes of this work.

7 Experimental Results

To evaluate performance of our solutions, we implemented all four secure tests in the semi-honest setting and in this section provide evaluation of their runtimes. Our implementation used Shamir (3, 1) secret sharing and we utilized the PICCO compiler [60] for transforming high-level programs into their corresponding secure distributed implementations. PICCO produces C/C++ programs that utilize the GMP [61] library for the underlying large-number arithmetic and OpenSSL's [62] implementation of AES for protecting communication. For each of the experiments, the smallest suitable field size for secret sharing was determined based on the declared variables and operations on them as described in [60]. All arithmetic on secret shares is then performed over a field $\mathbb{F}_p$ with prime p of the determined size. We report the modulus size as the bitlength of p.

We used three machines with identical hardware to run the experiments (each machine running a single computational party). All machines had commodity four-core 3.2GHz Intel i5-3470 processors with Red Hat Linux 2.6.32 and were connected via a 1Gb/s LAN. Most programs used a single core with the exception of the Fisher test, where a higher degree of parallelism is possible (three computational threads were running in parallel). We executed each experiment 10 times and report the mean value.

For each of the HWE, LD, CATT and Fisher tests, we vary the value of N to demonstrate how this variable affects performance of the computation. Furthermore, we also vary the number M of SNPs or alleles for which each test run, with all M instances of each test being executed at the same time. Table 2 shows performance of all tests for varying N and M, where we separately measure performance of the CATT test with private and public (non-zero w_1 and w_2) weights. Note that the Fisher test is usually used for small datasets and therefore, we use smaller values of N for this test. A parallelized implementation of the Fisher test is significantly more memory-intensive for the same value of N than other tests.

As expected, Table 2 shows that performance of the tests grows linearly with the number of locations M. The time per location slightly decreases as M increases for moderate values of M, which is due to amortizing communication delays over a larger number of tests. When, however, M becomes large, running M tests in parallel no longer provides computational savings and it may be desirable to partition the task into batches of smaller size. Increasing the value of N has a small impact on the runtime as complexity of the computation has a logarithmic dependency on N. For typical values of M around 1,000, our tests take a couple of seconds. Clearly, this is a feasible solution to privacy-preserving execution of genome-wide association and linkage studies.

Test	N	Modulus size	Number M of alleles			
			10	100	1,000	10,000
HWE	200	98	0.042	0.321	3.21	32.5
	400	104	0.046	0.355	3.39	33.9
	800	110	0.047	0.361	3.64	36.3
	1600	116	0.051	0.374	3.87	38.9
LD	200	89	0.037	0.298	2.99	30.6
	400	94	0.040	0.313	3.08	31.9
	800	99	0.042	0.337	3.18	32.1
	1600	104	0.043	0.345	3.37	33.7
CATT with private weights	200	86	0.036	0.297	2.92	29.5
	400	91	0.039	0.295	2.98	30.7
	800	96	0.040	0.319	3.02	31.3
	1600	101	0.045	0.348	3.23	32.6
CATT with public weights	200	86	0.035	0.291	2.86	29.1
	400	91	0.039	0.298	2.99	30.7
	800	96	0.039	0.308	3.07	31.5
	1600	101	0.041	0.340	3.27	32.7
Fisher	100	67	0.108	0.979	9.78	98.1
	200	68	0.217	2.09	20.9	N/A
	400	69	0.453	4.47	44.6	N/A

Table 2: Execution time for all tests measured in seconds.

If we compare our performance to that reported in [48], we can do that for the case of CATT (the only overlapping test). [48] used the Sharemind framework [63] with 3 computational servers utilizing 2 cores each for the computation (compared to 1 core in our experiments) and connected via a 1Gb/s LAN connection. The CATT test executed in parallel on all 262,264 SNPs in their dataset took on the order of 60 seconds for data coming from several hundred donors. While this is faster than in our experiments, there are notable differences in the setup. First, the Sharemind is highly optimized with the particular emphasis on running many instances of the same operation in parallel. Performance of many operations does not scale linearly with the batch size until the saturation point is reached (which for many integer operations is on the order of million operations), which means that a direct comparison of the numbers in this work with a single batch size in [48] is not accurate and puts us at a disadvantage for the target batch size of 1000 SNPs. Lastly, Sharemind's setup is more constrained compared to the standard secret sharing techniques (e.g., does not support more than 3 computational parties) and we expect to see similar performance for our tests if Sharemind was used. Aside from the performance comparison, we mention that the goal of this work was to provide a comprehensive suite of statistical tests for GWAS and GWLS, most of which were not covered in [48].

8 Conclusions

In this work, we show how a suite of statistical tests used in genome-wide association and linkage studies can be efficiently realized in the secure computation framework. It enables privacy-preserving distributed evaluation of such tests with provable security guarantees over highly sensitive genomic data. We cover all of HWE, LD, CATT, and Fisher tests to cover the entire chain of tests used in genome-wide studies. We hope that this line of work will encourage the use of secure computation techniques in genomic applications and opens new possibilities which presently remain unexplored due to the risk of data disclosure.

Acknowledgments

This work was supported in part by grants CNS-1223699 and CNS-1319090 from the National Science Foundation and FA9550-13-1-0066 from the Air Force Office of Scientific Research. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agencies.

References

1. Gurwitz D. and McLeod H. L. Genome-wide association studies: Powerful tools for improving drug safety and efficacy. *Pharmacogenomics*, 10(2), 2009.

2. Jin G. and Wong S. TC. Toward better drug repositioning: Prioritizing and integrating existing methods into efficient pipelines. *Drug Discovery Today*, 19(5):637–644, 2014.

3. Hurle MR., Yang L., Xie Q., Rajpal DK., Sanseau P. and Agarwal P. Computational drug repositioning: From data to therapeutics. *Clinical Pharmacology & Therapeutics*, 93(4):335–341, 2013.

4. Wapinski O. and Chang H. Y. Long noncoding RNAs and human disease. *Trends in Cell Biology*, 21(6):354–361, 2011.

5. Franke A., Balschun T., Karlsen T. H., Hedderich J., May S., Lu T. et al. Replication of signals from recent studies of Crohn's disease identifies previously unknown disease loci for ulcerative colitis. *Nature Genetics*, 40 (6):713–715, 2008.

6. Homer N., Szelinger S., Redman M., Duggan D., Tembe W., Muehling J. et al. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS Genetics*, 4(8), 2008.

7. Kolata G. Poking holes in genetic privacy. The New York Times (June 16, 2013). http://www.nytimes.com/2013/06/18/science/poking-holes-in-the-privacy-of-dna.html.

8. R.Wang , Li Y. F., Wang X., Tang H. and Zhou X. Learning your identity and disease from research papers: information leaks in genome wide association study. In *ACM Conference on Computer and Communications Security (CCS)*, pages 534–544, 2009.

9. Braun R., Rowe W., Schaefer C., Zhang J. and Buetow K. Needles in the haystack: Identifying individuals present in pooled genomic data. *PLoS Genet*, 5(10), 2009.

10. Jacobs K., Yeager M., Wacholder S., Craig D., Kraft P., Hunter D. et al. A new statistic and its power to infer membership in a genome-wide association study using genotype frequencies. *Nature Genetics*, 41(11):1253–1257, 2009.

11. Masca N., Burton P. and Sheehan N. Participant identification in genetic association studies: improved methods and practical implications. *International Journal of Epidemiology*, 40(6):1629–1642, 2011.

12. Schadt E., Woo S. and Hao K. Bayesian method to predict individual SNP genotypes from gene expression data. *Nature Genetics*, 44(5):603–608, 2012.

13. Gymrek M., McGuire A., Golan D., Halperin E. and Erlich Y. Identifying personal genomes by surname inference. *Science*, 339(6117):321–324, 2013.

14. Humbert M., Ayday E., Hubaux J.P. and Telenti A. Addressing the concerns of the Lacks family: quantification of kin genomic privacy. In *ACM Conference on Computer and Communications Security (CCS)*, pages 1141–1152, 2013.

15. Ghodsi M., Amiri S., Hassani H. and Ghodsi Z. An enhanced version of Cochran-Armitage trend test for genome-wide association studies. *Meta Gene*, 9:225–229, 2016.

16. Chen Z., Huang H. and Ng H. K. T. Testing for association in case-control genome-wide association studies with shared controls. *Statistical methods in medical research*, 25(2):954–967, 2016.

17. Lacour A., Schüller V., Drichel D., Herold C., Jessen F., Leber M. et al. Novel genetic matching methods for handling population stratification in genome-wide association studies. *BMC bioinformatics*, 16(1):1, 2015.

18. Nakajima M., Takahashi A., Tsuji T., Karasugi T., Baba H., Uchida K. et al. A genome-wide association study identifies susceptibility loci for ossification of the posterior longitudinal ligament of the spine. *Nature genetics*, 46(9):1012–1016, 2014.

19. Erlich Y. and Narayanan A. Routes for breaching and protecting genetic privacy. *Nature Reviews Genetics*, 15(6): 409–421, 2014.

20. Cai R., Hao Z., Winslett M., Xiao X., Yang Y., Zhang Z. et al. Deterministic identification of specific individuals from GWAS results. *Bioinformatics*, 31(11):1701–1707, 2015.

21. Agrawal R., Kiernan J., Srikant R. and Xu Y. Hippocratic databases. In *International Conference on Very Large Data Bases (VLDB)*, pages 143–154, 2002.

22. Agrawal R., Bayardo R., Faloutsos C., Kiernan J., Rantzau R. and Srikant R. Auditing compliance with a hippocratic database. In *International Conference on Very Large Data Bases (VLDB)*, pages 516–527, 2004.

23. Samarati P. and Sweeney L. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical Report SRI-CSL-98-04, Computer Science Laboratory, SRI International, 1998.

24. El Emam K. and Dankar F. K. Protecting privacy using k-anonymity. *Journal of the American Medical Informatics Association*, 15(5):627–637, 2008.

25. Malin B. Protecting genomic sequence anonymity with generalization lattices. *Methods of Information in Medicine*, 44:687–692, 2005.

26. Jha S., Kruger L. and Shmatikov V. Towards practical privacy for genomic computation. In *IEEE Symposium on Security and Privacy*, pages 216–230, 2008.

27. Bruekers F., Katzenbeisser S., Kursawe K. and Tuyls P. Privacy-preserving matching of DNA profiles. IACR Cryptology ePrint Archive 2008/203, 2008.

28. Katz J. and Malka L. Secure text processing with applications to private DNA matching. In *ACM Conference on Computer and Communications Security (CCS)*, pages 485–492, 2010.

29. Baldi P., Baronio R., Cristofaro E. De, Gasti P. and Tsudik G. Countering GATTACA: Efficient and secure testing of fully-sequenced human genomes. In *ACM Conference on Computer and Communications Security (CCS)*, pages 691–702, 2011.

30. Hormozdiari F., Joo J. W. J., Wadia A., Guan F., Ostrovsky R., Sahai A. et al. Privacy preserving protocol for detecting genetic relatives using rare variants. *Bioinformatics*, 30(12):i204–i211, 2014.

31. Blanton M. and Bayatbabolghani F. Efficient server-aided secure two-party function evaluation with applications to genomic computation. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2016(4), 2016.

32. De Cristofaro E., Faber S. and Tsudik G. Secure genomic testing with size and position hiding private substring matching. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 107–118, 2013.

33. Yasuda M., Shimoyama T., Kogure J., Yokoyama K. and Koshiba T. Secure pattern matching using somewhat homomorphic encryption. In *ACM Workshop on Cloud Computing Security*, pages 65–76, 2013.

34. Du W. and Atallah M. J. Secure multi-party computation problems and their applications: a review and open problems. In *Workshop on New Security Paradigms*, pages 13–22, 2001.

35. Atallah M. J. and Du W. Secure multi-party computational geometry. In *Workshop on Algorithms and Data Structures (WADS)*, pages 165–179. 2001.

36. Atallah M. J., Kerschbaum F. and Du W. Secure and private sequence comparisons. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 39–44, 2003.

37. Atallah M. J. and Li J. Secure outsourcing of sequence comparisons. *International Journal of Information Security*, 4(4):277–287, 2005.

38. Blanton M., Atallah M. J., Frikken K. B. and Malluhi Q. Secure and efficient outsourcing of sequence comparisons. In *European Symposium on Research in Computer Security (ESORICS)*, pages 505–522, 2012.

39. Lauter K., López-Alt A. and Naehrig M. Private computation on encrypted genomic data. In *Progress in Cryptology – LATINCRYPT*, pages 3–27, 2014.

40. Lu W., Yamada Y. and Sakuma J. Efficient secure outsourcing of genome-wide association studies. In *IEEE Security and Privacy Workshops (SPW)*, pages 3–6, 2015.

41. Wang S., Zhang Y., Dai W., Lauter K., Kim M., Tang Y. et al. HEALER: Homomorphic Computation of ExAct Logistic rEgRession for secure rare disease variants analysis in GWAS. *Bioinformatics*, 32(2):211–218, 2016.

42. Bos J. W., Lauter K. and Naehrig M. Private predictive analysis on encrypted medical data. *Journal of Biomedical Informatics*, 50:234–243, 2014.

43. Lu W., Yamada Y. and Sakuma J. Privacy-preserving genome-wide association studies on cloud environment using fully homomorphic encryption. *BMC Medical Informatics and Decision Making*, 15(Suppl 5):S1, 2015.

44. Kim M. and Lauter K. Private genome analysis through homomorphic encryption. *BMC Medical Informatics and Decision Making*, 15(Suppl 5):S3, 2015.

45. Zhang Y., Dai W., Jiang X., Xiong H. and Wang S. Foresee: Fully outsourced secure genome study based on homomorphic encryption. *BMC Medical Informatics and Decision Making*, 15(Suppl 5):S5, 2015.

46. Constable S. D., Tang Y., Wang S., Jiang X. and Chapin S. Privacy-preserving GWAS analysis on federated genomic datasets. *BMC Medical Informatics and Decision Making*, 15(Suppl 5):S2, 2015.

47. Zhang Y., Blanton M. and Almashaqbeh G. Secure distributed genome analysis for GWAS and sequence comparison computation. *BMC Medical Informatics and Decision Making*, 15(Suppl 5):S4, 2015.

48. Kamm L., Bogdanov D., Laur S. and Vilo J. A new way to protect privacy in large-scale genome-wide association studies. *Bioinformatics*, 29(7):886–893, 2013.

49. Shamir A. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

50. Catrina O. and De Hoogh S. Improved primitives for secure multiparty integer computation. In *Security and Cryptography for Networks (SCN)*, pages 182–199. 2010.

51. Goldreich O. and Ostrovsky R. Software protection and simulation on oblivious RAMs. *Journal of the ACM (JACM)*, 43(3):431–473, 1996.

52. Canetti R. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

53. Gennaro R., Rabin M. and Rabin T. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 101–111, 1998.

54. Hirt M. and Maurer U. Robustness for free in unconditional multi-party computation. In *Advances in Cryptology – CRYPTO*, pages 101–118, 2001.

55. Beerliova-Trubiniova Z. and Hirt M. Perfectly-secure MPC with linear communication complexity. In *Theory of Cryptography Conference (TCC)*, pages 213–230, 2008.

56. Damgård I., Ishai Y., Krøigaard M., Nielsen J. and Smith A. Scalable multiparty computation with nearly optimal work and resilience. In *Advances in Cryptology – CRYPTO*, pages 241–261, 2008.

57. Damgård I., Ishai Y. and Krøigaard M. Perfectly secure multiparty computation and the computational overhead of cryptography. In *Advances in Cryptology – EUROCRYPT*, pages 445–465, 2010.

58. Peng K. and Bao F. An efficient range proof scheme. In *IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT)*, pages 826–833, 2010.

59. Blanton M. and Aguiar E. Private and oblivious set and multiset operations. *To appear in International Journal of Information Security*, 2016.

60. Zhang Y., Steele A. and Blanton M. Picco: A general-purpose compiler for private distributed computation. In *ACM Conference on Computer and Communications Security (CCS)*, pages 813–826, 2013.

61. GMP – The GNU multiple precision arithmetic library. http://www.gmplib.org.

62. OpenSSL: The open source toolkit for SSL/TLS. http://www.openssl.org.

63. Bogdanov D., Laur S. and Willemson J. Sharemind: A framework for fast privacy-preserving computations. In *European Symposium on Research in Computer Security (ESORICS)*, pages 192–206, 2008.