Cryptographic Solutions for Credibility and Liability Issues of Genomic Data

Erman Ayday*[1], Qiang Tang**[2], and Arif Yilmaz[1]

[1]Computer Engineering Department, Bilkent University, Ankara, Turkey
[2]University of Luxembourg, L-1359 Luxembourg

Abstract. In this work, we consider a scenario that includes an individual sharing his genomic data (or results obtained from his genomic data) with a service provider. In this scenario, (i) the service provider wants to make sure that received genomic data (or results) in fact belongs to the corresponding individual (and computed correctly), (ii) the individual wants to provide a digital consent along with his data specifying whether the service provider is allowed to further share his data, and (iii) if his data is shared without his consent, the individual wants to determine the service provider that is responsible for this leakage. We propose two schemes based on homomorphic signatures and aggregate signatures that link the information about the legitimacy of the data to the consent and the phenotype of the individual. Thus, to verify the data, each party also needs to use the correct consent and phenotype of the individual who owns the data.

1 Introduction

With the rapid decrease in the cost of whole genome sequencing and genotyping, today, genomic data is widely used in healthcare, research, and even in recreational genomics. However, benefits due to this wide use of genomic data come along with potential threats against individuals' privacy. Genomic data of an individual includes privacy-sensitive data about him such as his physical characteristics, predisposition to diseases, and family members. Therefore, it is crucial to protect privacy of an individual's genomic data while allowing him to utilize his data to receive certain healthcare or recreational services. As a result, there has been significant amount of research efforts on privacy-preserving processing

* Erman Ayday is supported by MSCA Individual Fellowship from the European Commission and by the Scientific and Technological Research Council of Turkey, TUBITAK, under Grant No. 115E766.

** Qiang Tang is supported by a CORE grant from the National Research Fund, Luxembourg and an internal project from University of Luxembourg.

and secure storage of genomic data. However, the credibility and liability issues on genomic data have not been widely considered in the literature.

Lots of individuals share their (anonymized) genomic data for research purposes. Such donations are very important for the research community as researchers need large amounts of genomic data samples to increase the statistical power of their studies. Similarly, some service providers make computations on genomic data of individuals and they are only interested in the results of such computations (rather than the raw genomic data). However, researchers (or service providers) want to make sure that either (i) a donated genome indeed belongs to a particular individual, or (ii) the results of a genetic test is indeed computed from the correct data of the particular individual. In this work, we study this credibility issue and propose cryptographic techniques that would enable a researcher (or a service provider) to verify the credibility of a donated genome (or a computed genomic test).

Furthermore, as an individual donates his genomic data for research (to a particular entity) or undergoes a genetic test from a service provider, he would like to make sure that neither his genomic data nor his genomic test results are going to be observed by other individuals. Privacy leakage occurs when genomic data of the individual or his genomic test results are publicly shared by the service providers that collect such data at the first place. In such incidents, it is important to understand whom to keep liable due to such a leakage. Thus, (i) the individual wants to provide a digital consent along with his data specifying whether the service provider is allowed to further share his data, and (ii) if his data is shared without his consent, the individual wants to determine the service provider that is responsible from this leakage.

Our main assumption is that the service provider (which receives genomic data or genomic test results from an individual) should prove the legitimacy of the data when sharing it with other entities. Under this assumption, if the service provider makes the data public (without the consent of the individual), it will be detected by the individual. Similarly, if the service provider tries to share the data offline with another (non-malicious) entity, that entity will understand that the corresponding data is being shared without the consent of the data owner. Note however that if the unauthorized offline sharing of genomic data is between a malicious service provider and other malicious service providers, there is no technical solution to detect this leakage.

1.1 Contribution

In a nutshell, we propose two schemes to share genomic data and genetic test results, respectively. The proposed schemes are based on both homomorphic signature and aggregate signature that links the information about the legitimacy of the data to the consent and the phenotype (or the identity) of the individual. Thus, in order to verify the data, a party also needs to use the correct consent and phenotype of the individual who owns the data.

One proposed scheme allows the service providers to check the validity of individuals' genomic data. The other proposed scheme allows service providers to conduct genetic tests on individuals' data and be assured that the test is conducted accurately. The adoption of homomorphic signature enables the individual to honestly share any subset of the authenticated data or the test results without interacting with the authority. Moreover, it guarantees that the individual does not leak unnecessary information when sharing the test results. The adoption of aggregate signature efficiently prevents illegal (or unauthorized) sharing of genomic data by the service providers. In such a case, either the entity which receives the data understands that data is shared without the consent of the data owner, or the data owner can understand which service provider leaked his data without his consent, and hence he can hold that party liable of the leakage.

We emphasize that the proposed schemes can be easily adopted by existing works on privacy-preserving processing of genomic data in order to have a complete pipeline.

1.2 Organization

The rest of the paper is organized as follows. In the next section, we discuss the related work on security/privacy of genomic data and content ownership techniques. In Section 3, we briefly provide background information on homomorphic signatures, aggregate signatures, and genomics. In Section 4, we introduce our system and threat models. In Section 5, we provide the details of the solution for sharing genomic data along with the security analysis. In Section 6, we describe the protocol for sharing the results of a genetic test. In Section 7, we discuss the security properties of the solution and evaluate the practicality of the proposed scheme. Finally, in Section 8, we conclude the paper.

2 Related Work

There have been several works on security and privacy of genomic data. However, as mentioned, credibility and liability issues of genomic data have not been considered in previous work. We briefly summarize the existing efforts on security/privacy of genomic data in the following.

One line of investigation is represented by works focusing on private clinical genomics. Baldi et al. presented efficient algorithms for privacy-preserving testing on full genomes, including paternity and ancestry testing, and the testing of point mutations (single nucleotide polymorphisms - SNPs) for partner compatibility and personalized medicine [4]. Ayday et al. proposed a scheme to protect the privacy of users' genomic data yet enable medical units to access the genomic data in order to conduct medical tests or to develop personalized medicine methods [2]. Karvelas et al. proposed using the oblivious RAM mechanisms to access genomic data (that is stored at a third party) and secure two-party computation protocols to compute various functionalities on the data [15]. Recently, Wang et al. proposed private edit distance protocols to find similar patients (e.g., across several hospitals) [19]. To provide secure storage and retrieval of genomic data, Ayday et al. proposed techniques for the privacy-preserving storage and retrieval of raw-genomic data [1], and Huang et al. proposed a scheme that would guarantee long-term security (in an information-theoretical sense) for genomic data [11].

Another area of interest addresses the problem of protecting genomic privacy and still allowing for both basic and translational medical research on the data. It has been shown that standard anonymization techniques are ineffective on genomic data [9]. It has also been shown that the identity of a participant of a genomic study can be revealed by using a second sample, that is, part of the DNA information from the individual and the results of the corresponding clinical study [10]. Furthermore, Humbert et al. evaluated the genomic privacy of an individual threatened by his or her relatives revealing their genomes [12]. As a response to these threats, a few solutions have been proposed. These can be put in three main categories: (i) techniques based on differential privacy, in which a controlled noise is added to the result of a query (to a genomic database) [13], (ii) techniques based on cryptography, in which the use of homomorphic encryption, secure hardware, or secure multiparty computation are proposed for privacy-preserving genomic research [14, 8], and (iii) techniques based on optimization, in which the goal is to maximize the amount of

publicly shared genomic data and also comply to the privacy preferences of individuals.

There have also been many attempts to prove the credibility (or authenticity) of a given message or document. The most common tools to provide this functionality are digital signatures [17]. Digital signatures are widely used for software distribution, financial transactions, and in other cases in which it is important to detect forgery or tampering. However, using a digital signature to prove the credibility of a genome has two main disadvantages: (i) digital signature can reveal the identity of the genome donor, and (ii) genomic data is usually shared or donated partially, but the signature is typically computed over the whole data at the data generator side (e.g., sequencing facility). On the other hand, liability issues of a digital content are typically addressed by using a watermarking technique on the document [3]. However, (i) digital watermarking techniques are proved to be functional for multimedia content, but not for informative text, (ii) watermarking techniques typically include injecting some level of noise to the data, which might not be tolerated for health-related data, and (iii) a watermark is typically included on the whole file (e.g., image), but genomic data can be partially shared.

3 Preliminaries

In this section, we provide background information for homomorphic and aggregate signatures (which are the main building blocks of our proposed technique) and genomics in general.

3.1 Signature Schemes

Homomorphic signatures. Similar to homomorphic encryption scheme which enables computation on encrypted data, homomorphic signature scheme enables computation on signed data. Suppose a user Alice has a set of messages $\{m_1, \cdots, m_k\}$. She can (independently) sign each data element and store the signatures at a cloud server. Later, Alice can ask the server to compute authenticated functions of the signed data (e.g., a signature for the mean value of the messages), solely based on the individual signatures. Given the mean value and the signature from the server, any user can verify the signature. Many homomorphic signature schemes have been proposed in the literature, as surveyed in [18]. Next, we briefly introduce the linearly homomorphic signature scheme $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Evaluate})$ from [6] that we will use in this work. The scheme is detailed in Appendix A.

- $\mathsf{Setup}(1^n, k)$. On input a security parameter n and a data set size k, this algorithm outputs a public/private key pair $(\mathsf{pk^h}, \mathsf{sk^h})$. The parameter k defines how many signatures can be involved in the homomorphic operation. The message space is $\mathbb{F}_p^n$, where p is a prime number, and signatures are short vectors in $\mathbb{Z}^n$. A function $f \in \mathcal{F}$ is encoded as $\langle f \rangle = (c_1, \cdots, c_k) \in \mathbb{Z}^k$, where $\mathcal{F}$ includes all $\mathbb{F}_p$-linear functions on k-tuples of messages in $\mathbb{F}_p$.
- $\mathsf{Sign}(\mathsf{sk^h}, \tau, m, i)$. On input a secret key $\mathsf{sk^h}$, a tag $\tau \in \{0,1\}^n$, a message $m \in \mathbb{F}_p^n$, and an index i, this algorithm outputs a signature σ. Note that τ can be considered as an identifier of the dataset that m belongs to, while i is the index of m in this dataset.
- $\mathsf{Verify}(\mathsf{pk^h}, \tau, m, \sigma, f)$. On input a public key $\mathsf{pk^h}$, a tag $\tau \in \{0,1\}^n$, a message $m \in \mathbb{F}_p^n$, a signature $\sigma \in \mathbb{Z}^n$, and a function $f \in \mathcal{F}$, this algorithm outputs 1 (accept) or 0 (reject).
- $\mathsf{Evaluate}(\mathsf{pk^h}, \tau, f, \overrightarrow{\boldsymbol{\sigma}})$. On input a public key $\mathsf{pk^h}$, a tag $\tau \in \{0,1\}^n$, a function $f \in \mathcal{F}$ encoded as $\langle f \rangle = (c_1, \cdots, c_k) \in \mathbb{Z}^k$, and a tuple of signatures $\overrightarrow{\boldsymbol{\sigma}} = (\sigma_1, \cdots, \sigma_k) \in \mathbb{Z}^n$, the algorithm outputs $\sigma = \sum_{i=1}^k c_i \sigma_i$.

Two security properties are defined for homomorphic signature: *unforgeability* and *context-hiding*. Informally, the unforgeability property implies that an attacker will not be able to forge a signature for a new tag τ' (generated by the attacker himself). Moreover, the attacker will not be able to forge a signature for a message which is not equal to the evaluation of f on the existing signed messages. Suppose that $\overrightarrow{\boldsymbol{\sigma}} = (\sigma_1, \cdots, \sigma_k)$ are the signatures for messages in $\{m_1, \cdots, m_k\}$ with respect to a tag τ, then $\mathsf{Verify}(\mathsf{pk^h}, \tau, m, \sigma, f) = 0$ if $m \neq \sum_{i=1}^k f_i m_i$. The context-hiding property implies that the signature σ, output of $\mathsf{Evaluate}$, does not leak more information about $\{m_1, \cdots, m_k\}$ than $\sum_{i=1}^k f_i m_i$.

Aggregate signatures. To improve the efficiency of cascaded sharing of SNPs and test results, we also use aggregate signatures. Suppose there are N users, denoted as $\{U_1, \ldots, U_N\}$, of an aggregate signature scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Aggregate})$. Suppose that each user U_i, with the key pair $(\mathsf{pk^a}_i, \mathsf{sk^a}_i)$, generates a signature $\sigma_i = \mathsf{Sign}(\mathsf{sk^a}_i, m_i)$ for message m_i. Then, given σ_i $(1 \leq i \leq N)$ values from all users, any entity can run $\mathsf{Aggregate}$ to aggregate them into a single signature σ_{agg}. With $\mathsf{pk^a}_i, m_i$ $(1 \leq i \leq N)$ and σ_{agg}, any entity can verify whether these signatures are valid or not. In this paper, we use the Boneh-Lynn-Shacham aggregate signature scheme [7], which achieves standard unforgeability property. The scheme is detailed in Appendix B.

3.2 Genomics Background

The human genome is encoded in double stranded DNA molecules consisting of two complementary polymer chains. Each chain consists of simple units called nucleotides (A,C,G,T). Even though most of the DNA sequence is conserved across the whole human population, around 0.5% of each person's DNA (which corresponds to several millions of nucleotides) is different from the reference genome, owing to genetic variations. Single nucleotide polymorphism (SNP) is the most common DNA variation. A SNP is a position in the genome holding a nucleotide that varies between individuals and there are approximately 4 million SNPs in each individual. Multiple Genome Wide Association Studies (GWAS) performed in recent years have shown that a patient's susceptibility to particular diseases can be (partially) predicted from sets of his SNPs. Thus, leakage of SNPs often poses a significant threat to individual privacy.

Each SNP position includes two alleles (i.e., two nucleotides) and everyone inherits one allele of every SNP position from each of his parents. If an individual receives the same allele from both parents, he is said to be homozygous for that SNP position. If, however, he inherits a different allele from each parent (one minor and one major), he is called heterozygous. Depending on the alleles the individual inherits from his parents, the content of a SNP position can be simply represented as the number of minor alleles it possesses, i.e., 0, 1, or 2. A service provider may run various linear tests on the SNPs of an individual. For example, a service provider may compute the predicted susceptibility of patient P for disease X, $\mathbb{S}_P^X$, by using weighted averaging [2] as follows:

$$\mathbb{S}_P^X = \sum_{i \in \varphi_X} \mathrm{w}^i_{\mathrm{SNP}_i^P}(X) \times \mathrm{SNP}_i^P, \tag{1}$$

where, φ_X includes the indices of SNPs that are relevant for disease X and $\mathrm{w}_j^i(X)$ represents the contribution of different states of SNP j (i.e., 0, 1, or 2) for disease X.

4 System and Security Models

Here we describe the system model, threat model, and the initialization for the proposed scheme.

4.1 The System Model

We assume the existence of multiple certified institutions (CIs), individuals, and service providers (SPs) in the system. For the sake of simplicity,

we will describe the proposed scheme using a single CI, individual (Alice), and SP. Our proposed system model is also illustrated in Fig. 1.

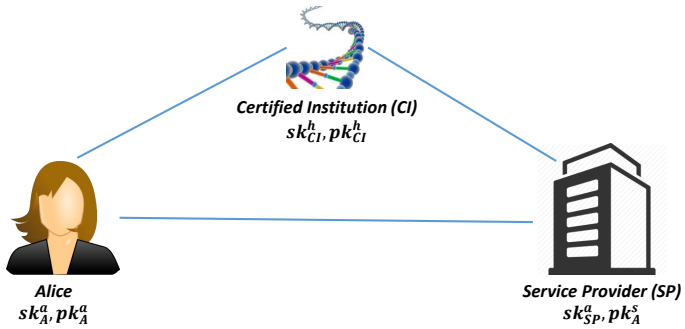

Fig. 1. Proposed System Model

The CI is mainly responsible for sequencing, encrypting, and signing the sequenced data. In this work, we do not consider encryption at the CI, as it is not the main focus of the paper. However, there has been several works in the literature that cover such encryption techniques. Our proposed scheme can easily be adopted by one of such schemes to provide a complete pipeline. Furthermore, it is worth noting that a certified institution for sequencing has been proposed in many existing works on genomic privacy [2]. Having such a CI is also unavoidable in today's sequencing technology. In practice, the SP can be a medical institution, a genetic researcher, or a direct-to-customer (DTC) service provider. The SP is mainly interested in receiving a portion of Alice's genome (e.g., for research) or the result of a (linear) genetic test that is conducted on Alice's genome. It has been shown that the results of such genetic tests are particularly important to determine (i) the predisposition of an individual for different diseases, or (ii) the exact dose of a drug that will be prescribed to an individual. Alice, on the other hand, is interested in either (i) enrolling in a genetic research initiative by donating a part of her genome (e.g., a subset of her SNPs), (ii) sharing a part of her genome with a medical institution for treatment, or (iii) receiving a service based on the result of a genetic test that will be run on her genome. In all these scenarios, Alice wants to share her data either anonymously (without her real identity) or with her real identity. Furthermore, she also wants to provide a consent denoting whether the SP can further share the genomic

data it received from Alice with other entities (either anonymously or with the real identity of Alice).

When the system is set up, we assume the following keys have been generated and certified by a certificate authority (CA).

- The CI generates a key pair $(\mathsf{sk}^{\mathsf{h}}{}_{CI}, \mathsf{pk}^{\mathsf{h}}{}_{CI})$ for the Boneh-Freeman homomorphic signature scheme. During the key generation, the CI should set the parameters according to the pre-defined sequencing tasks. Suppose the set of SNPs for Alice is $\mathbf{G}$ with the size $|\mathbf{G}|$, then the k parameter should be $|\mathbf{G}|+2$, required by the proposed protocols. The parameter p should be selected such that it makes Equality (2), defined in Section 5.1, hold with very small probability.
- Alice generates a key pair $(\mathsf{sk}^{\mathsf{a}}{}_{A}, \mathsf{pk}^{\mathsf{a}}{}_{A})$ for the Boneh-Lynn-Shacham aggregate signature scheme.
- The SP generates a key pair $(\mathsf{sk}^{\mathsf{a}}{}_{SP}, \mathsf{pk}^{\mathsf{a}}{}_{SP})$ for the Boneh-Lynn-Shacham aggregate signature scheme.

As a standard practice, we assume the CA generates a certificate for every public key and is responsible for all maintenance issues. For simplicity, we omit the details here. With respect to Alice's public key $\mathsf{pk}^{\mathsf{a}}{}_{A}$, we assume the associated certificate $Cert_A$ does not contain the Alice's real identity ID_A because we want to allow Alice to anonymously share her data (when desired). However, we require the CA to issue a specifical certificate $Cert_{pk-id-A}$ to link ID_A and $\mathsf{pk}^{\mathsf{a}}{}_{A}$.

4.2 The Threat Model

To be realistic and avoid single point of failure, we assume there are two trust anchors in the system. First, all parties trust the CA(s) to certify the public keys used to protect genomic data, as shown in Fig. 2. In reality, the CA(s) can be government agencies or entities endorsed by such agencies. We could even require the CI to be certified by more than one CAs. For simplicity, we assume there is only one CA in our discussion. Second, all parties trust the CI to generate genomic data (via sequencing) and link the generated data to individual users, as shown in Fig. 3.

Since we want to focus on the credibility and liability issues, we simply assume there are secure communication channels between all parties. Therefore, an outside attacker will neither learn the genomic data and test results (confidentiality) nor modify them (integrity). Under these assumptions, we mainly consider two types of attacks in our security evaluation.

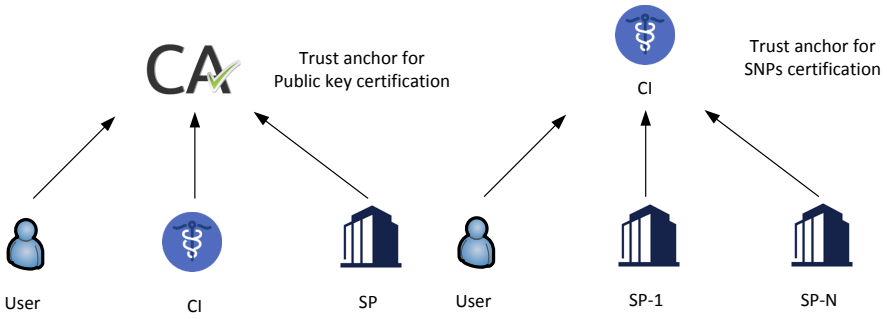

Fig. 2. Trust model between the certificate authority (CA), the user, the certified institution (CI), and the service provider (SP).

Fig. 3. Trust model between the certified institution (CI), the user, and the service provider (SP).

- *Credibility attack.* A malicious party (e.g., a user or SP) may try to provide modified genomic data or test results in participating in genomic research. In practice, a user may provide fake genomic data or test results to get compensation from the government, and a malicious SP may forward modified genomic data or test results to another SP to mislead its research.
- *Liability attack.* A malicious party (e.g., a SP or CI) may try to forge a user's consent in order to share his/her genomic data or test results with another *honest party*. As mentioned before, if two malicious parties want to share a user's data at their hands, we do not have technical way to stop it and should resort to other countermeasures.

4.3 Initialization

We have two message formats in the proposed scheme representing the SNPs and the consent.

- The message format of SNP i of Alice is denoted as an n-tuple $M_i^s = (ID_A, g_i, 0, \cdots, 0)$, where ID_A is the Alice's identity and g_j is the value of SNP j ($j \in \mathbf{G}$ and $g_j \in \{0, 1, 2\}$). The $(n-2)$ 0s in M_i^s are to meet the message format of the Boneh-Freeman homomorphic signature scheme.
- The message format of consent is represented as $M^c = (ID_A || C_{A,SP}(t) || ID_{SP})$, where ID_{SP} is the identity of the SP for the corresponding transaction, and $C_{A,SP}(t)$ represents the actual consent. In its simplest form, $C_{A,SP}(t)$ can be {"do not share", "share anonymously", "share non-anonymously"}, and can be defined freely.

After the setup, Alice and the CI interact as follows for Alice to register at the CI.

1. Alice sends her identity ID_A, her public key $\mathsf{pk^a}_A$ and associated certificate $Cert_A$, and $Cert_{pk-id-A}$ to the CI.
2. The CI validates the following facts: Alice owns the phonotype P_A, the certificate $Cert_A$ for $\mathsf{pk^a}_A$ is correct, and $Cert_{pk-id-A}$ is valid and links ID_A and $\mathsf{pk^a}_A$. If the validation passes, the CI selects $\tau_A \in \{0,1\}^n$ and sends it to Alice. Note that n is the security parameter of the Boneh-Freeman homomorphic signature scheme. At the end, the CI establishes a record $ID_A, P_A, \mathsf{pk^a}_A, Cert_A, Cert_{pk-id-A}, \tau_A$ for Alice.

At any time, Alice can send her biological sample to the CI, which will then sequence her genome and sign the results. In more detail, Alice and the CI perform the following protocol shown in Fig. 4.

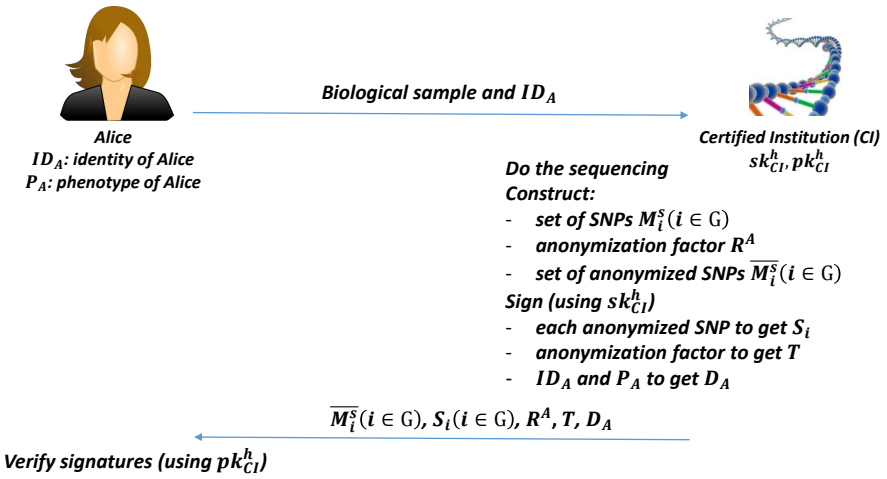

Fig. 4. Initialization and genome sequencing between Alice and the CI.

– Step 1: Alice sends her biological sample along with ID_A and P_A to the CI.
– Step 2: The CI does the sequencing and determines the SNPs in $\mathbf{G}$.
– Step 3: The CI constructs $M_i^s = (ID_A, g_i, 0, \cdots, 0)$ for each SNP, where $i \in \mathbf{G}$.
– Step 4: The CI selects the anonymization factor $R^A = (\ell^A, 0, 0, \cdots, 0)$ where $\ell_A \xleftarrow{\$} \mathbb{Z}_p$ which means ℓ_A is chosen from $\mathbb{Z}_p$ uniformly at ran-

dom. Note that p is the security parameter of the Boneh-Freeman homomorphic signature scheme. The anonymization factor is used when Alice wants to share her data anonymously.

- Step 5: The CI constructs anonymized SNPs $\overline{M_i^s} = (ID_A - \ell^A, g_i, 0, \cdots, 0)$ for every $i \in \mathbf{G}$.
- Step 6: The CI signs each anonymized SNP message using homomorphic signature scheme and $\mathsf{sk^h}_{CI}$ to obtain $S_i = \mathsf{Sign}(\mathsf{sk^h}_{CI}, \tau_A, \overline{M_i^s}, i)$ for every $i \in \mathbf{G}$.
- Step 7: The CI signs the anonymization factor R^A to obtain $T = \mathsf{Sign}(\mathsf{sk^h}_{CI}, \tau_A, R^A, |\mathbf{G}| + 1)$.
- Step 8: The CI signs the ID of Alice along with her phenotype information to obtain $D_A = \mathsf{Sign}(\mathsf{sk^h}_{CI}, \tau_A, (ID_A, P_A, 0, \cdots, 0), |\mathbf{G}| + 2)$.
- Step 9: The CI sends anonymized SNPs, corresponding signatures (i.e. S_i values), the anonymization factor (i.e. R^A), T, and D_A to Alice.
- Step 10: Alice verifies all received signatures.

To facilitate the following discussions, we define a message vector $\overrightarrow{\boldsymbol{M}}$ and a signature vector $\overrightarrow{\boldsymbol{\sigma}}$ with $|\mathbf{G}| + 2$ elements as follows.

$$\overrightarrow{\boldsymbol{M}} = (\overline{M_1^s}, \cdots, \overline{M_{|\mathbf{G}|}^s}, R^A, (ID_A, P_A, 0, \cdots, 0)), \overrightarrow{\boldsymbol{\sigma}} = (S_1, \cdots, S_{|\mathbf{G}|}, T, D_A)$$

5 Protocol for Sharing SNPs

If Alice wants to share her SNPs with the SP, they engage in the protocol shown in Fig. 5. We sketch the protocol in the following, and leave the details to Appendix C.

- Step 1: The SP sends the indices of the SNPs it requests, denoted by $\mathbf{I} = \{i_1, \cdots, i_t\}$.
- Step 2: Alice retrieves the corresponding anonymized SNPs $\overline{M_j^s}$ ($j \in \mathbf{I}$) along with the corresponding anonymity factor R^A.
- Step 3: Alice generates $|\mathbf{G}| + 2$ random coefficients to construct a function f which has the encoding form $\langle f \rangle := (f_1, \cdots, f_{|\mathbf{G}|+2})$. The generation of f is detailed below.

  Let PF be a Hash function, which outputs $|\mathbf{G}| + 2$ numbers $r_1, \cdots, r_{|\mathbf{G}|+2}$. When Alice generates $\langle f \rangle = (f_1, \cdots, f_{|\mathbf{G}|+2}) \in \mathbb{Z}^{|\mathbf{G}|+2}$, she first generates $r_1, \cdots, r_{|\mathbf{G}|+2}$ using $\mathsf{pk}_A||\mathsf{pk}_{SP}||\tau_A||i_1||\cdots||i_t||\overline{M_{i_1}^s}||\cdots||\overline{M_{i_t}^s}||R^A$ as input. Then, she sets $f_{i_j} := r_{i_j}$ for every requested SNP in $\mathbf{I}$, sets $f_{|\mathbf{G}|+1} := r_{|\mathbf{G}|+1}, f_{|\mathbf{G}|+2} := r_{|\mathbf{G}|+2}$, and sets $f_x = 0$ for other x (i.e.,

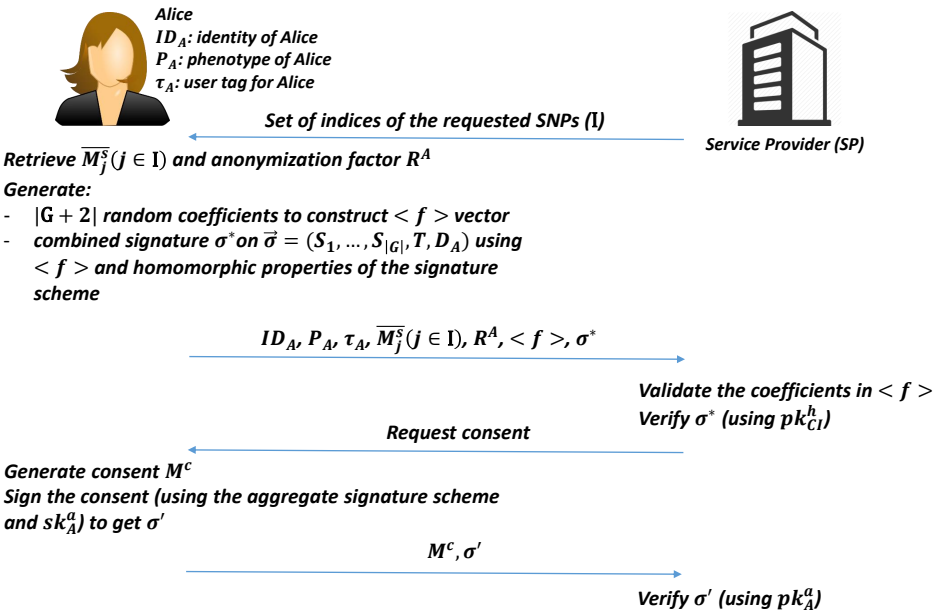

Fig. 5. Non-anonymous SNP Sharing.

for the SNPs that are not in $\mathbf{I}$). Thus, any entity, including the SP, can validate $\langle f \rangle$ is generated in this manner.

- Step 4: Alice generates a combined signature using the homomorphic properties of the digital signature scheme. $\sigma^* := \mathsf{Evaluate}(\mathsf{pk^h}_{CI}, \tau_A, f, \vec{\boldsymbol{\sigma}})$, where $\vec{\boldsymbol{\sigma}} = (S_1, \cdots, S_{|\mathbf{G}|}, T, D_A)$.
- Step 5: Alice sends ID_A, P_A, τ_A, $\overline{M_j^s}$ values $(j \in \mathbf{I})$, R^A, $\langle f \rangle$, and σ^* to the SP.
- Step 6: The SP validates $\langle f \rangle$ (as coefficients in $\langle f \rangle$ are publicly verifiable) and verifies σ^*.
- Step 7: The SP requests the consent from Alice.
- Step 8: Alice generates the consent M^c and signs it using her private key to obtain $\sigma' := \mathsf{Sign}(\mathsf{sk^a}_A, M^c||\tau_A||info)$, where $info := i_1||\cdots||i_t||\overline{M_{i_1}^s}||\cdots||\overline{M_{i_t}^s}||R^A$.
- Step 9: Alice sends M^c and σ' to the SP.
- Step 10: The SP verifies the signature. The use of aggregate signature for further re-sharing of the same data (assuming Alice has consent for re-sharing) is further discussed in Section 5.1.

5.1 Security Analysis

In the proposed protocol, the generation of challenge $\langle f \rangle$ plays a key role in preventing credibility attacks. We discuss two cases.

- *Alice tries to cheat SP.* In this case, some of the SNPs information from Alice, namely $\overline{M_{i_j}^s}(1 \leq j \leq t)$ and R^A, is different from what has been signed by the CI. The unforgebility property of the homomorphic signature scheme guarantees that $\langle f \rangle \cdot \overrightarrow{\boldsymbol{M}}^T$ is computed correctly by Alice, and the corresponding signature σ^* is valid. Otherwise, we will have a forgery for the signature scheme. As such, Alice can only successfully mount an attack when the following equality holds.

$$\langle f \rangle \cdot \overrightarrow{\boldsymbol{M}^*}^T = \langle f \rangle \cdot \overrightarrow{\boldsymbol{M}}^T, \tag{2}$$

  where

$$\overrightarrow{\boldsymbol{M}^*} = ((0,0,0,\cdots,0),\cdots,\overline{M_{i_1}^s},\cdots,\overline{M_{i_t}^s},\cdots,R^A,(ID_A,P_A,0,\cdots,0))$$

  Based on the generation of f, it is straightforward to show that the equality holds with negligible probability with reasonable parameters, so that it is infeasible for Alice to mount the attack. Furthermore, the consent signature σ' also prevents this malicious behavior as σ' includes the contents and indices of the requested SNPs along with the consent.
- *Alice colludes with SP to cheat another SP.* This scenario is exactly the same as the above scenario. Due to the fact that the generation of $\langle f \rangle$ is publicly verifiable, collusion does not give Alice any additional advantage.

The unforgebility property of the Boneh-Lynn-Shacham aggregate signature scheme guarantees that the SP has been authorized by Alice to use SNPs and has the privileges specified in the consent M^c. The $info$ parameter links the signature σ' to the shared SNPs data.

Suppose that $SP^{(0)}$ has been authorized by Alice to further share her SNPs data. If $SP^{(0)}$ wants to share the SNPs with $SP^{(1)}$ then it will generate a signature $\sigma'_{0\to1}$ for a consent of the form $M^c||\tau_A||info||ID_{SP^{(1)}}$. Similarly, $SP^{(1)}$ can generate a signature $\sigma'_{1\to2}$ for a consent of the form $M^c||\tau_A||info||ID_{SP^{(2)}}$ to share the SNPs with $SP^{(2)}$. This process can continue, and form a chain of delegated consents: $\sigma'_{0\to1}$, $\sigma'_{1\to2}$, $\cdots$, $\sigma'_{N-1\to N}$. $SP^{(N)}$ can aggregate the signatures into a single one

$\sigma'_{0\to1\to\cdots\to N}$. When $SP^{(N)}$ wants to share Alice's data with Bob, it provides the following information.

$$\sigma^*, f, ID_A, P_A, M^c||\tau_A||info, ID_{SP^{(0)}}, \cdots, ID_{SP^{(N)}}, \sigma'_{0\to1\to\cdots\to N}$$

Bob can then validate all the signatures in the chain to see whether $SP^{(N)}$ has obtained the permission or not. Moreover, Bob can validate the SNPs data by validating σ^*. Note that the SNPs data can be obtained from the $info$ parameter.

5.2 Anonymous Sharing

In order to stay anonymous, Alice follows the same protocol, shown in Fig. 5, except that she does not transmit R^A, ID_A, and P_A to the SP. When Alice generates $\langle f \rangle$, she should set $f_{|\mathbf{G}|+1} := 0, f_{|\mathbf{G}|+2} := 0$. The security analysis remains the same.

6 Protocol for Sharing Test Results

If Alice wants to share the genetic test results with the SP, they engage in the protocol shown in Fig. 6. We sketch the protocol in the following, and leave the details to Appendix D.

- Step 1: The SP sends the weights of the test $(w_1, \cdots, w_{|\mathbf{G}|})$ to Alice (to be general, we assume all SNPs to be used in the test).
- Step 2: Alice constructs the first $|\mathbf{G}|$ values of $\langle f \rangle$ based on the weights and sets $f_{|\mathbf{G}|+1} := f_{|\mathbf{G}|+2} := 0$.
- Step 3: Alice computes the result of the test m^* using her SNPs and the received weights.
- Step 4: Alice generates a combined signature σ^* using the homomorphic properties of the digital signature scheme. $\sigma^* := \mathsf{Evaluate}(\mathsf{pk^h}_{CI}, \tau_A, f, \overrightarrow{\boldsymbol{\sigma}})$, where $\overrightarrow{\boldsymbol{\sigma}} = (S_1, \cdots, S_{|\mathbf{G}|}, T, D_A)$.
- Step 5: Alice also constructs her consent M^c and signs it to generate $\sigma' := \mathsf{Sign}(\mathsf{sk^a}_A, M^c||\tau_A||info||m^*)$, where $info := w_1||\cdots||w_{|\mathbf{G}|}$.
- Step 6: Alice sends m^*, σ^*, σ', τ_A, and M^c to the SP.
- Step 7: The SP verifies both signatures it receives from Alice.

If Alice wants to share her phenotype information P_A with the SP, the she can send ID_A, P_A and D_A to the SP. By doing so, Alice loses anonymity.

The unforgebility property of the homomorphic signature scheme guarantees that the test result m^* is faithfully computed based on Alice's

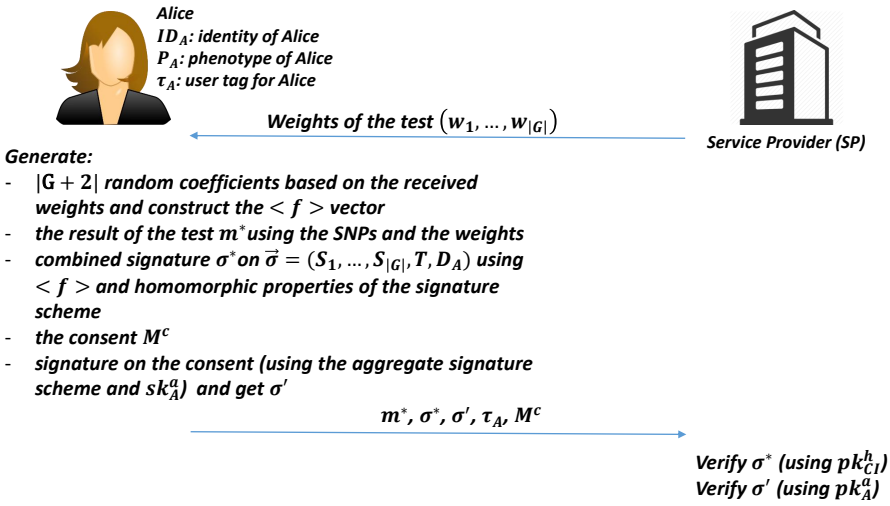

Fig. 6. Anonymous Test Result Sharing.

data, while the context hiding property guarantees that the signature σ^* does not leak more information than m^* about Alice's SNPs. The unforgeability property of the aggregate signature scheme guarantees that the SP has been authorized by Alice to use test results and has the privileges specified in the consent. If the test results are going to be shared further with other SPs, the workflow is the same as that of sharing SNPs.

7 Discussion

In general, all signatures (on data, ID, and phenotype) are generated by the CI. Using the homomorphic properties of the digital signature scheme (as discussed in Section 3.1), Alice linearly combines such signatures (depending on the type of the query) and generates a valid signature that can be verified by using the public key of the CI. As discussed in Section 5.1, Alice cannot cheat an SP by providing incorrect SNP data.

We assume that the SP, when sharing Alice's data with other entities, needs to show proof that the data is legitimate. This proof is the digital signature that SP receives from Alice (signed using the aggregate signature scheme and Alice's private key). As discussed, the signature can only be verified by using the correct consent of Alice. Therefore, the SP will be detected if it tries to share Alice's data without her consent. A malicious SP may try to modify the consent of Alice in order to share her data with other entities (along with a valid signature). However, since the consent

is signed by Alice's private key at the first place, such an attack is also not possible.

A malicious SP may also publicly share Alice's SNP data without her consent. We assume that such a sharing also includes the signature to prove the credibility of the shared data. In such a scenario, the $\langle f \rangle$ values in the corresponding signature would reveal the identity of the malicious SP that leaked Alice's data without her consent. This property of the proposed scheme brings a solution for the liability issues on case of unauthorized sharing of genomic data (since the values in $\langle f \rangle$ are generated using the public key of the SP, as discussed in Section 5).

One drawback of the proposed scheme is that it does not prevent an SP from linking the anonymous identity of Alice to her real identity. Assume Alice shares a set of SNPs with a particular SP in a non-anonymous way. Then, if Alice shares another set of SNPs on a public database in an anonymous way, the SP can deanonymize Alice's identity as it possesses the R_A value of Alice from the previous transaction. We will further study this issue in future work.

We also briefly remark on the performance of the proposed solutions. First, we recap the implementation results of the Boneh-Lynn-Shacham aggregate signature scheme due to Barreto et al. [5]. Suppose that the implementation is based on a super-singular curve. For a computer with PIII 1 GHz CPU, signing takes 3.57 milliseconds, while verification takes 53 milliseconds. The aggregation algorithm Aggregate only incurs multiplications in the source group, and each multiplication takes less than 14 microseconds. Verifying an aggregate signature with k individual signatures takes roughly $53 \cdot k$ milliseconds.

Second, we remark on the homomorphic signature scheme. At this moment, we do not have any available implementation for this primitive, however this does not prevent us from estimating its performances. The most costly function is the Sign algorithm, whose main complexity comes from the SamplePre routine which is basically a sampling algorithm for Gaussian distribution. According to the implementation of Lyubashevsky and Prest [16], based on an Intel Core i5-3210M laptop with a 2.5GHz CPU and 6GB RAM, a Gaussian sampling takes about 115 milliseconds. The Verify and Evaluate algorithms are much more efficient because they only incur linear operations and has no exponentiations. As future work, we will build a proof-of-concept prototype and have the precise performance numbers.

8 Conclusions

In this work, we proposed two cryptographic schemes to share genomic data and genetic test results. The proposed schemes are between a data owner and a service provider. Using the proposed schemes, on the one hand, a service provider can check the validity (or legitimacy) of genomic data it receives from a data owner (individual). On the other hand, the individual, via a digital consent, can make sure that the service provider will not further share his data without his permission. The proposed schemes are based on homomorphic signatures and aggregate signatures, and these cryptographic primitives enable us to link the information about the legitimacy of the data to the consent and the identity of the individual. We also discussed the security and practicality of the proposed schemes. The proposed schemes can be easily adopted by existing works on privacy-preserving processing of genomic data.

References

1. Ayday, E., Raisaro, J.L., Hengartner, U., Molyneaux, A., Hubaux, J.P.: Privacy-preserving processing of raw genomic data. In: DPM (2013)
2. Ayday, E., Raisaro, J.L., Rougemont, J., Hubaux, J.P.: Protecting and evaluating genomic privacy in medical tests and personalized medicine. In: WPES (2013)
3. A.Z.Tirkel, Rankin, G., Schyndel, R.V., W.J.Ho, N.R.A.Mee, C.F.Osborne: Electronic water mark. In: DICTA 93. pp. 666–673 (1993)
4. Baldi, P., Baronio, R., De Cristofaro, E., Gasti, P., Tsudik, G.: Countering GATTACA: Efficient and secure testing of fully-sequenced human genomes. Proceedings of ACM CCS '11 pp. 691–702 (2011)
5. Barreto, P.S.L.M., Kim, H.Y., Lynn, B., Scott, M.: Advances in cryptology — crypto 2002. pp. 354–369. Springer Berlin Heidelberg (2002)
6. Boneh, D., Freeman, D.M.: Homomorphic signatures for polynomial functions. In: Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology. pp. 149–168. Springer-Verlag (2011)
7. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: A survey of two signature aggregation techniques. CryptoBytes 6(2), 1–9 (2003)
8. Canim, M., Kantarcioglu, M., Malin, B.: Secure management of biomedical data with cryptographic hardware. IEEE Transactions on Information Technology in Biomedicine 16(1) (2012)
9. Gymrek, M., McGuire, A.L., Golan, D., Halperin, E., , Erlich, Y.: Identifying personal genomes by surname inference. Science: 339 (6117) (Jan 2013)
10. Homer, N., Szelinger, S., Redman, M., Duggan, D., Tembe, W.: Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. PLoS Genetics 4 (Aug 2008)
11. Huang, Z., Ayday, E., Hubaux, J.P., Fellay, J., , Juels, A.: Genoguard: Protecting genomic data against brute-force attacks. In: n Proceedings of IEEE Symposium on Security and Privacy (2015)

12. Humbert, M., Ayday, E., Hubaux, J.P., Telenti, A.: Addressing the concerns of the Lacks family: Quantification of kin genomic privacy. In: CCS (2013)
13. Johnson, A., Shmatikov, V.: Privacy-preserving data exploration in genome-wide association studies. In: KDD. pp. 1079–1087 (2013)
14. Kantarcioglu, M., Jiang, W., Liu, Y., Malin, B.: A cryptographic approach to securely share and query genomic sequences. IEEE Transactions on Information Technology in Biomedicine 12(5), 606–617 (2008)
15. Karvelas, N., Peter, A., Katzenbeisser, S., Tews, E., Hamacher, K.: Privacy-preserving whole genome sequence processing through proxy-aided oram. In: Proceedings of the 13th Workshop on Privacy in the Electronic Society. pp. 1–10 (2014)
16. Lyubashevsky, V., Prest, T.: Quadratic time, linear space algorithms for gram-schmidt orthogonalization and gaussian sampling in structured lattices. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. Lecture Notes in Computer Science, vol. 9056, pp. 789–815. Springer (2015)
17. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21(2), 120–126 (Feb 1978)
18. Traverso, G., Demirel, D., Buchmann, J.: Homomorphic Signature Schemes: A Survey. SpringerBriefs in Computer Science, Springer (2016)
19. Wang, R., Wang, X., Li, Z., Tang, H., Reiter, M.K., Dong, Z.: Privacy-preserving genomic computation through program specialization. Proceedings of the 16th ACM Conference on Computer and Communications Security pp. 338–347 (2009)

Appendix A Boneh-Freeman Signature Scheme

The Boneh-Freeman homomorphic signature scheme is based on lattices, and we recap the description here. The reader should refer to [6] for more details.

- $\mathsf{Setup}(1^n, k)$. On input a security parameter n and a data set size k, do the following:
  1. Choose two primes $p, q = \mathsf{poly}(n)$ with $q \geq (nkp)^2$. Define $\ell := \lfloor n/6 \log q \rfloor$.
  2. Set $\Lambda_1 := p\mathbb{Z}_n$.
  3. Use $\mathsf{TrapGen}(q, \ell, n)$ to generate a matrix $\mathbf{A} \in \mathbb{F}_q^{\ell \times n}$ along with a short basis $\mathbf{T}_q$ of $\Lambda_q^{\perp}(\mathbf{A})$. Define $\Lambda_2 := \Lambda_q^{\perp}(\mathbf{A})$ and $\mathbf{T} := p \cdot \mathbf{T}_q$. Note that $\mathsf{TrapGen}$ is a function to sample matrices in lattices.
  4. Set $\upsilon := p \cdot \sqrt{n \log q} \cdot \log n$
  5. Let $H : \{0,1\}^* \to \mathbb{F}_q^{\ell}$ be a hash function.
  6. Output the public key $\mathsf{pk}^{\mathsf{h}} := (\Lambda_1, \Lambda_2, \upsilon, k, H)$ and the secret key $\mathsf{sk}^{\mathsf{h}} := \mathbf{T}$.

  The public key pk^{h} defines the following system parameters:
  - The message space is $\mathbb{F}_p^n$ and signatures are short vectors in $\mathbb{Z}^n$.
  - The set of admissible functions $\mathcal{F}$ is all $\mathbb{F}_p$-linear functions on k-tuples of messages in $\mathbb{F}_p$.
  - For a function $f \in \mathcal{F}$ defined by $f(m_1, \cdots, m_k) = \sum_{i=1}^{k} c_i m_i$, we encode f by interpreting the c_i as integers in $(-p/2, p/2]$ and defining $\langle f \rangle = (c_1, \cdots, c_k) \in \mathbb{Z}^k$.
  - To evaluate the hash function ω_τ on an encoded function $\langle f \rangle = (c_1, \cdots, c_k) \in \mathbb{Z}^k$, do the following:
    1. For $i = 1, \cdots, k$, compute $\alpha_i = H(\tau || i) \in \mathbb{F}_q^{\ell}$
    2. Define $\omega_\tau(\langle f \rangle) = \sum_{i=1}^{k} c_i \alpha_i \in \mathbb{F}_q^{\ell}$.

    Note that ω_τ is a hash function that maps encoding of function f to elements of $\mathbb{Z}^n / \Lambda_2$.
- $\mathsf{Sign}(\mathsf{sk}^{\mathsf{h}}, \tau, m, i)$. On input a secret key sk^{h}, a tag $\tau \in \{0,1\}^n$, a message $m \in \mathbb{F}_p^n$, and an index i, do:
  1. Compute $\alpha_i = H(\tau || i) \in \mathbb{F}_q^{\ell}$. Then, by definition, $\omega_\tau(\langle \pi_i \rangle) = \alpha_i$.
  2. Compute $\mathbf{t} \in \mathbb{Z}^n$ such that $\mathbf{t} \mod p = m$ and $\mathbf{A} \cdot \mathbf{t} \mod q = \alpha_i$.
  3. Output $\sigma \leftarrow \mathsf{SamplePre}(\Lambda_1 \bigcap \Lambda_2, \mathbf{T}, \mathbf{t}, \upsilon) \in \Lambda_1 \bigcap \Lambda_2 + \mathbf{t}$.

  Note that $\mathsf{SamplePre}$ is basically a sampling algorithm for Gaussian distributions.

- $\mathsf{Verify}(\mathsf{pk}^{\mathsf{h}}, \tau, m, \sigma, f)$. On input a public key pk^{h}, a tag $\tau \in \{0,1\}^n$, a message $m \in \mathbb{F}_p^n$, a signature $\sigma \in \mathbb{Z}^n$, and a function $f \in \mathcal{F}$, If all of the following conditions hold, output 1 (accept); otherwise output 0 (reject).
  1. $||\sigma|| \leq k \cdot \frac{p}{2} \cdot \upsilon\sqrt{n}$.
  2. $\sigma \mod p = m$.
  3. $\mathbf{A} \cdot \sigma \mod q = \omega_\tau(\langle f \rangle)$.
- $\mathsf{Evaluate}(\mathsf{pk}^{\mathsf{h}}, \tau, f, \overrightarrow{\boldsymbol{\sigma}})$. On input a public key pk^{h}, a tag $\tau \in \{0,1\}^n$, a function $f \in \mathcal{F}$ encoded as $\langle f \rangle = (c_1, \cdots, c_k) \in \mathbb{Z}^k$, and a tuple of signatures $\overrightarrow{\boldsymbol{\sigma}} = (\sigma_1, \cdots, \sigma_k) \in \mathbb{Z}^n$, output $\sigma = \sum_{i=1}^{k} c_i \sigma_i$.

Appendix B Boneh-Lynn-Shacham Signature Scheme

A bilinear group generator is an algorithm $\mathcal{G}_{\mathcal{C}}$ that takes as input a security parameter λ and outputs a description $\Gamma = (p, \mathbb{G}, \mathbb{G}_T, \hat{e}, g)$ where:

- $\mathbb{G}$ and $\mathbb{G}_T$ are groups of prime order p with efficiently computable group laws.
- g is a randomly-chosen generator of $\mathbb{G}$.
- $\hat{e}$ is an efficiently-computable bilinear pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, i.e., a map satisfying the following properties for $g \neq 1 \in \mathbb{G}$:
  - Bilinearity: $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$ for all $a, b \in \mathbb{Z}_{pq}$;
  - Non-degeneracy: $\hat{e}(g, g) \neq 1$.

The Boneh-Lynn-Shacham aggregate signature scheme [7] are defined with four algorithms.

- $\mathsf{Setup}(\lambda)$. On input of the security parameter λ, this algorithm runs $\mathcal{G}_{\mathcal{C}}$ to generate $\Gamma = (p, \mathbb{G}, \mathbb{G}_T, \hat{e}, g)$, and generates a hash function $H : \{0,1\}^* \to \mathbb{G}$.
- $\mathsf{KeyGen}(\Gamma)$. This algorithm chooses $s \xleftarrow{\$} \mathbb{Z}_p$ and set the key pair to be $(\mathsf{pk^a}, \mathsf{sk^a})$ where $\mathsf{pk^a} = g^s$ and $\mathsf{sk^a} = s$.
- $\mathsf{Sign}(\mathsf{sk^a}, m)$. On input of the private key $\mathsf{sk^a}$ and a message m, this algorithm outputs the signature $\sigma = H(\mathsf{pk^a}||m)^s$.
- $\mathsf{Verify}(\mathsf{pk^a}, m, \sigma)$. On input of the public key $\mathsf{pk^a}$, a message m and its signature σ, the algorithm outputs 1 iff $\hat{e}(g, \sigma) = \hat{e}(H(\mathsf{pk^a}||m), \mathsf{pk^a})$.
- $\mathsf{Aggregate}(\Sigma)$. On input of a set of signatures $\Sigma = \{\sigma_i (1 \leq i \leq k)\}$, which are signed by $\mathsf{pk^a}_i$ for message m_i correspondingly, this algorithm outputs $\sigma_{agg} = \prod_{i=1}^{k} \sigma_i$.

With an aggregate signature, the verification outputs 1 iff $\hat{e}(g, \sigma_{agg}) = \prod_{i=1}^{k} \hat{e}(H(\mathsf{pk^a}_i||m), \mathsf{pk^a}_i)$.

Appendix C Protocol for Sharing SNPs Non-anonymously

SP		**Alice**
$(\mathsf{sk}^{\mathsf{h}}{}_{SP}, \mathsf{pk}^{\mathsf{h}}{}_{SP})$		$(\mathsf{sk}^{\mathsf{a}}{}_{A}, \mathsf{pk}^{\mathsf{a}}{}_{A})$
% identify the SNPs %		
$(i_1, \cdots, i_t)$		
	$\xrightarrow{ID_{SP}, (i_1, \cdots, i_t)}$	
		$\overline{M^s_{i_j}}\ (1 \le j \le t), R^A$
		% compute the signature %
		set $\langle f \rangle := (f_1, \cdots, f_{\lvert\mathbf{G}\rvert+2})$
		$\sigma^* := \mathsf{Evaluate}(\mathsf{pk}^{\mathsf{h}}{}_{CI}, \tau_A, f, \overrightarrow{\boldsymbol{\sigma}})$
	$\xleftarrow{\tau_A, ID_A, P_A}$	
	$\xleftarrow{\overline{M^s_{i_j}}\ (1 \le j \le t), R^A}$	
	$\xleftarrow{\langle f \rangle, \sigma^*}$	
validate $\langle f \rangle$		
$m^* := \langle f \rangle \cdot \overrightarrow{\boldsymbol{M}^*}^T$		
$\mathsf{Verify}(\mathsf{pk}^{\mathsf{h}}{}_{CI}, \tau_A, m^*, \sigma^*, f)$		
	"consent" $\xrightarrow{\qquad}$	
		% sign consent%
		$info := i_1 \lvert\lvert \cdots \lvert\lvert i_t \lvert\lvert \overline{M^s_{i_1}} \lvert\lvert \cdots \lvert\lvert \overline{M^s_{i_t}} \lvert\lvert R^A$
		$\sigma' := \mathsf{Sign}(\mathsf{sk}^{\mathsf{a}}{}_{A}, M^c \lvert\lvert \tau_A \lvert\lvert info)$
	$\xleftarrow{M^c, \sigma'}$	
% verify consent %		
$\mathsf{verify}(\mathsf{pk}^{\mathsf{a}}{}_{A}, M^c \lvert\lvert \tau_A \lvert\lvert info, \sigma')$		

Fig. 7. Non-anonymous SNPs Sharing

The vector $\overrightarrow{\boldsymbol{M}^*}$ is defined as follows.

$$\overrightarrow{\boldsymbol{M}^*} = ((0,0,0,\cdots,0), \cdots, \overline{M^s_{i_1}}, \cdots, \overline{M^s_{i_t}}, \cdots, R^A, (ID_A, P_A, 0, \cdots, 0))$$

Appendix D Protocol for Sharing Test Results Anonymously

SP		**Alice**
$(\mathsf{sk}_{SP}, \mathsf{pk}_{SP})$		$(\mathsf{sk}_A, \mathsf{pk}_A)$
% prepare test function %		
$(w_1, \cdots, w_{\lvert\mathbf{G}\rvert})$		
	$\xrightarrow{(w_1, \cdots, w_{\lvert\mathbf{G}\rvert})}$	
		% if sharing is ok %
		set $\langle f \rangle = (f_1, \cdots, f_{\lvert\mathbf{G}\rvert+2})$,
		$s.t.\ f_i := w_i$ for every $1 \leq i \leq \lvert\mathbf{G}\rvert$
		$f_{\lvert\mathbf{G}\rvert+1} := 0, f_{\lvert\mathbf{G}\rvert+2} := 0$
		$m^* := (f_1, \cdots, f_{\lvert\mathbf{G}\rvert+2}) \cdot \overrightarrow{\boldsymbol{M}}^T$
		% sign test results%
		$\sigma^* := \mathsf{Evaluate}(\mathsf{pk}^{\mathsf{h}}{}_{CI}, \tau_A, f, \overrightarrow{\boldsymbol{\sigma}})$
		% sign consent%
		$info := w_1 \lvert\lvert \cdots \lvert\lvert w_{\lvert\mathbf{G}\rvert}$
		$\sigma' := \mathsf{Sign}(\mathsf{sk}^{\mathsf{a}}{}_A, M^c \lvert\lvert \tau_A \lvert\lvert info \lvert\lvert m^*)$
	$\xleftarrow{m^*, \sigma^*, \sigma', \tau_A, M^c}$	
$\mathsf{Verify}(\mathsf{pk}^{\mathsf{h}}{}_{CI}, \tau_A, m^*, \sigma^*, f)$		
$\mathsf{Verify}(\mathsf{pk}^{\mathsf{a}}{}_A, M^c \lvert\lvert \tau_A \lvert\lvert info, \sigma')$		

Fig. 8. Anonymous Test Result Sharing