Privacy-preserving Disease Susceptibility Test with Shamir's Secret Sharing

Guyu Fan[1] and Manoranjan Mohanty[2]
[1] *New York University Abu Dhabi, Abu Dhabi, U.A.E.*
[2] *Center for Cyber Security, New York University Abu Dhabi, Abu Dhabi, U.A.E.*

Keywords: Shamir's Secret Sharing, Genome Based Disease Susceptibility Test, Single Nucleotide Polymorphism (SNP).

Abstract: Recent advances in genomics have facilitated the development of personalized medicine, in which a patient's susceptibility to certain diseases and her compatibility with certain medications can be determined from her genetic makeup. Although this technology has many advantages, privacy of the patient is one of the major concerns due to the sensitivity of genomic data. In this paper, we propose a privacy-preserving scheme for computing a patient's susceptibility to a particular disease. Our scheme stores genomic data in hidden form and performs the disease susceptibility test in the hidden domain. To hide the data, we use Shamir's (l,n) secret sharing, which can be homomorphic to a fixed number of multiplications and unlimited additions. Using Shamir's secret sharing, we create n shares and store the shares at n datacenters. The datacenters perform the susceptibility test on their shares and send the result (which is also hidden) to a hospital. Finally, the hospital obtains the secret result of the test by accessing at least k datacenters, where $k = 2l - 1$. In comparison to other works, our approach is more practical as it minimizes the involvement of the patient and incurs less overhead.

1 INTRODUCTION

The success of the Human Genome Project and the rapid increase of computational power have resulted in a variety of applications of genomic data (Naveed et al., 2015). For example, in health care, doctors can now compute a patient's susceptibility to a certain disease using the patient's genome data. Although this genome-based disease susceptibility test (DST) (Butts et al., 2016) is cost-effective and mostly accurate, privacy of the patient is a major concern (Naveed et al., 2015) (Ayday et al., 2013).

A patient's genomic data can reveal various sensitive information of the patient and her blood relatives. For instance, disease information derived from genomic data can be used for discrimination, and inferred physical features such as skin color may be used to identify the individual. Moreover, the privacy implications of genomic data is not fully known yet as the scientific community is far from completely understanding the human genome. Therefore, genomic data must be protected from unauthorized access using tight security measures and legislations (Naveed et al., 2015).

Hospitals often lack the expertise to protect their patients' genomic data (Ayday et al., 2013). Due to the large size (several gigabytes per person) of the data (Naveed et al., 2015), it is usually difficult for hospitals to securely store, process, and maintain the genomic data of patients when they have limited resources. For instance, hospitals may not have the latest technologies or adequate manpower to successfully prevent new hacking attempts (Ayday et al., 2013). A possible solution to this problem is to outsource the storage and processing of genomic data to a third-party service provider (*e.g.,* a cloud datacenter) in a privacy-preserving way (Ayday et al., 2013). The service provider can better handle the data by providing more efficient and more secure storage and processing technologies. It is, however, imperative that the service provider not know the data in plaintext. Based on third-party outsourcing, Ayday *et al.* proposed the first privacy-preserving DST scheme that stores encrypted genomic data in a third-party server (Ayday et al., 2013). Their scheme uses Paillier encryption to encrypt genomic data, which allows them to perform encrypted domain DST using the weighted-averaging method (Kathiresan et al., 2008). However, their scheme is not practically suitable as it incurs high overhead and requires patients to be well educated in the field of genomics. Although certain follow-up studies (Danezis and De Cristofaro,

Fan, G. and Mohanty, M.
Privacy-preserving Disease Susceptibility Test with Shamir's Secret Sharing.
DOI: 10.5220/0006467105250533
In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT*, pages 525-533
ISBN: 978-989-758-259-2

2014) (Djatmiko et al., 2014) (Barman et al., 2015) have built upon Ayday *et al.*'s scheme, they still require further improvement (details will be discussed in Section 2).

In this paper, we propose a privacy-preserving DST framework based on Shamir's secret sharing (Shamir, 1979). Our scheme incurs less overhead and minimizes the participation of a patient in the process. Using Shamir's (l,n) secret sharing, we create n shares of the genomic data, and distribute the shares to n datacenters. Any group of less than l datacenters will learn nothing about the secret data. Our approach allows one multiplication and unlimited additions of the shared (*i.e.,* encrypted) values. Unlike Ayday *et al.*'s scheme, we do not store additional multiplied values, therefore the data overhead is reduced. To further improve efficiency, we outsource the DST computations from the hospital to the datacenters, which usually have more computational resources. Finally, to prevent the datacenters from inferring the disease from the DST, we camouflage the DST computation by introducing dummy genomic data (*i.e.,* operands) and dummy SNP weights. In our experimental setup, our scheme runs $10,000$ times faster than Ayday *et al.*'s scheme.

The rest of this paper is organized as follows. Section 3 provides background information on genomics and Shamir's secret sharing. Section 2 presents related work. In Section 4, we discuss our framework based on secret sharing, and in Section 5, we present analysis and experimental results. Section 6 concludes our work.

2 RELATED WORK

2.1 Ayday *et al.*'s Scheme

The main idea behind Ayday *et al.*'s scheme (Figure 1) (Ayday et al., 2013) is to store the SNPs in a third-party Storage and Processing Unit (SPU, *i.e.,* a datacenter) in encrypted form, and allow a medical center (MC, *i.e.,* a hospital) to access part of the SNPs. Given the biological sample of a patient, a trusted entity called Certified Institution (CI) first sequences the sample to obtain the patient's SNPs in digital form, and then encrypts the states and positions of the SNPs. The state of a SNP is encrypted using Paillier double encryption scheme (Bresson et al., 2003), whereas the position is encrypted using symmetric encryption. The encrypted SNPs are stored in the SPU. When an MC wants to perform a DST, it sends the location of the required SNPs to the patient. The patient checks whether the MC has permission to access the SNPs, and if so, sends the encrypted SNP positions (encrypted using the same symmetric encryption scheme) to the SPU. The SPU fetches the SNPs, re-encrypts the SNPs using the modified Paillier cryptosystem, and sends the encrypted SNPs to the MC. The MC performs the weighted-averaging-based DST (*i.e.,* computes S_P^X) on encrypted SNPs, and sends the encrypted result to the SPU. The SPU partially decrypts the result and sends it to the MC, where the result is fully decrypted.

However, Ayday *et al.*'s scheme has several practical issues. Firstly, the use of the modified Paillier cryptosystem results in high storage and computation overhead as a 2-bit SNP state (*i.e.,* 0, 1, or 2) will be represented as an 8192-bit ciphertext-pair (because 2048-bit keys are recommended for the Paillier cryptosystem). Secondly, since the Paillier cryptosystem is not homomorphic to multiplications, the CI must pre-compute the squared values of SNP states and store the squared values at the SPU. Finally, patients are actively involved in this scheme, which is generally undesirable. Not only do they have to perform symmetrical encryptions with their smartcard, they also need to be knowledgeable about genomics to decide if the MC's SNP requests are legitimate. The participation of the patient is both user-unfriendly and insecure as a wrong decision by the patient can leak sensitive information (and the patient will be responsible).

2.2 Other Schemes

After the seminal work of Ayday *et al.*, several studies have been carried out to improve and extend Ayday *et al.*'s scheme.

In (Danezis and De Cristofaro, 2014), Danezis *et al.* proposed a SNP-encoding scheme that eliminates the need for ciphertext multiplications. They used the faster El-Gamal cryptosystem instead of the Paillier cryptosystem. Although the overhead is decreased, the patient is still required to participate in the test. The patient also needs to store the encryption keys in a smart card, which when lost could cause a security breach. Furthermore, Danezis *et al.*'s scheme discloses the number of SNPs to a third-party server. Compared to their scheme, our keyless scheme is more secure as it completely eliminates the need for a smartcard, and hides the number of SNPs from the server using data obfuscation.

Djatmiko *et al.* (Djatmiko et al., 2014) proposed a Paillier-based scheme that can securely store and compute linear combinations of genomic data on a user's mobile device. Similar to Ayday *et al.*'s scheme, their scheme also incurs high overhead.

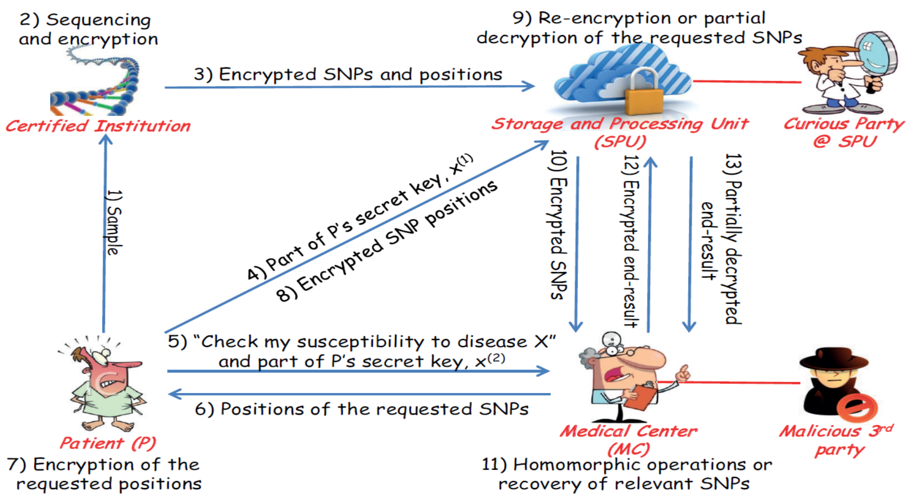
2) Sequencing and encryption
Certified Institution
3) Encrypted SNPs and positions
9) Re-encryption or partial decryption of the requested SNPs
Storage and Processing Unit (SPU)
Curious Party @ SPU
1) Sample
4) Part of P's secret key, $x^{(1)}$
8) Encrypted SNP positions
10) Encrypted SNPs
12) Encrypted end-result
13) Partially decrypted end-result
5) "Check my susceptibility to disease X" and part of P's secret key, $x^{(2)}$
6) Positions of the requested SNPs
Patient (P)
7) Encryption of the requested positions
Medical Center (MC)
11) Homomorphic operations or recovery of relevant SNPs
Malicious 3rd party

Figure 1: Ayday *et al.*'s privacy-preserving DST protocol(Ayday et al., 2013).

Moreover, storing and computing genomic data at the user-end can be less secure than in a datacenter.

In addition to private genome-based DST, significant work has also been done in related areas. For example, Huang *et al.* (Huang et al., 2015) presented a *honey encryption*-based genomic data storage framework that can protect data from brute-force attacks. Jha *et al.*, (Jha et al., 2008) presented a privacy-preserving way to perform dynamic-programming-based sequence alignment. Karvelas *et al.* (Karvelas et al., 2014) proposed an Oblivious RAM-based scheme that can securely store and compute genomic data in a third-party server.

3 PRELIMINARIES

3.1 Genome-based DST

Genome-based DST uses the SNPs (Single Nucleotide Polymorphism) of an individual's genome (Johnson and O'Donnell, 2009) (Ayday et al., 2013). A SNP is a position in the genome that holds a nucleotide that varies from one person to another. There are approximately 4 million SNPs in an individual's genome, and there are approximately 50 million total SNPs across the human population. An *allele* (*i.e.,* nucleotide) in a SNP can either occur frequently or infrequently in the human SNP pool. The frequent allele is referred to as the major allele, and the infrequent minor. Since a SNP contains one allele from each parent, it has three possible states: state 1 (exactly one minor allele), state 2 (two minor alleles), and state 0 (no minor alleles). To carry out a DST, it is sufficient to know the states and positions of the relevant SNPs (Ayday et al., 2013).

One of the advanced methods of computing disease susceptibilities is the *weighted-averaging* method (Kathiresan et al., 2008) (Ayday et al., 2013). This method is discussed in Ayday *et al.* and will be considered in this paper, too. Below is a description of this method.

Let SNP_i^P represent the state of SNP_i for patient P at position i, where $\text{SNP}_i^P \in \{0,1,2\}$. Suppose that the susceptibility of disease X is determined by a set of SNPs whose positions are given in the set L_X. Each such SNP has a contribution to disease X based on its state. More specifically, these contributions are state-sensitive probabilities as follows: $p_0^i(X) = \Pr(X|\text{SNP}_i^P = 0)$, $p_1^i(X) = \Pr(X|\text{SNP}_i^P = 1)$, and $p_2^i(X) = \Pr(X|\text{SNP}_i^P = 2)$. In addition, each SNP also has a general, state-agnostic contribution to disease X, and the general contribution of SNP_i is denoted C_i^X. Together, the C_i^X's and $p_j^i(X)$'s are called the SNP *weights* for disease X. Given C_i^X's and $p_j^i(X)$'s for each SNP_i, patient P's susceptibility to disease X can be computed as:

$$S_P^X = \frac{1}{\sum_{i\in L_X} C_i^X} \times \sum_{i\in L_X} C_i^X \left[\frac{p_0^i(X)}{(0-1)(0-2)}(\text{SNP}_i - 1) \right.$$
$$(\text{SNP}_i - 2) + \frac{p_1^i(X)}{(1-0)(1-2)}(\text{SNP}_i - 0)(\text{SNP}_i - 2)$$
$$\left. + \frac{p_2^i(X)}{(2-0)(2-1)}(\text{SNP}_i - 0)(\text{SNP}_i - 1) \right]. \qquad (1)$$

Note that in S_P^X, one SNP state is multiplied by another SNP state exactly once.

3.2 Shamir's Secret Sharing

Shamir's (l,n) secret sharing (Shamir, 1979) is a well established cryptosystem that hides a secret S by creating n shares of the secret such that less than l shares will not reveal any information about S. The shares are distributed among n participants (*e.g.,* datacenters), and it is assumed that no group of l or more participants will collude.

To create shares from S, we first pick a prime number p such that $p > S$. Then we define a $(l-1)$-degree polynomial

$$F(x) = (S + \alpha_x) \bmod p, \tag{2}$$

where $\alpha_x = \sum_{i=1}^{l-1} a_i x^i$ and $a_i < p$ is a random number in GF(p). Finally, using this polynomial, the q^{th} share of S is generated by setting $x = q$ and sent to the q^{th} participant.

To restore the secret S from the shares, at least l shares $\{z_0, z_1, \dots z_{l-1}\}$ such that $z_i = F(x_i)$ are required. Using these shares and their corresponding share numbers $\{x_0, x_1, \dots x_{l-1}\}$, the polynomial $F(x)$ is reconstructed using Lagrange interpolation as $L(x) = \sum_{i=0}^{l-1} z_i m_i(x) \bmod p$, where $m_i(x) = \prod_{j=0, j\neq i}^{m-1} \frac{x-x_j}{x_i-x_j}$ is the Lagrange basis function. S is found by setting $x = 0$ in $L(x)$ (which is equivalent to $F(x)$).

Shamir's secret sharing is homomorphic to additions and scalar multiplication. If the participants are holding shares of a set of secrets $S = \{S_1, S_2, ..., S_j\}$, then without communicating amongst themselves, they can compute the shares of the secret $\sum_{i=1}^{j} I_i S_i$, where I_i is an integer. Using this property, a number of privacy-preserving medical imaging schemes have been proposed (Mohanty et al., 2012) (Mohanty et al., 2013b) (Mohanty et al., 2013a).

However, we would like to note that with slight modifications, Shamir's secret sharing can also be homomorphic to a fixed number of multiplications (Gennaro et al., 1998). If the participants are holding shares of a set of secrets $S = \{S_1, S_2, ..., S_j\}$, then without communicating amongst themselves, they can compute the shares of the secret $\prod_{i=1}^{j} S_i$, where j is a pre-determined number. If the operand shares (i.e., shares of S_i's) are obtained from a $(l-1)$-degree polynomial, then the resultant share (i.e., shares of $\prod_{i=1}^{j} S_i$) will be a $j(l-1)$-degree polynomial. Therefore, to obtain the secret $\prod_{i=1}^{j} S_i$, at least $j(l-1)+1$ distinct shares will be required. In our case, one share (*i.e.,* SNP state) will be multiplied by another share only once. Thus, we need at least $2l-1$ shares to obtain the secret.

4 OUR APPROACH

In this section, we present our privacy-preserving DST scheme. We employ Shamir's secret sharing to hide a patient's SNPs and store the hidden SNPs in a number of datacenters. Upon request from the hospital, the datacenters perform an obfuscated version of the weighted-averaging-based DST computation in the encrypted domain without any involvement of the patient. With the encrypted DST results collected from these datacenters, the hospital obtains the final result in plaintext. We discuss the architecture and workflow of our scheme below.

4.1 Architecture

As shown in Figure 2, our framework contains 4 parties (a) a patient, (b) a trusted entity (TE), (c) n datacenters, and (d) a hospital. Our threat model assumes that (i) the patient is trusted, *i.e.,* the patient honestly performs all required operations and does not leak her genomic information; (ii) the TE is trusted; (iii) the datacenters are honest but curious, *i.e.,* they perform their operations honestly but can be curious to know the secret information from the DST (*e.g.,* by analyzing the data they operate on); and (iv) the hospital is usually honest but curious, although it can also be a malicious entity that deliberately deviates from the protocol in order to learn unauthorized information. Finally, we assume that communications between different parties (*e.g.,* between the patient and the TE) are secured, and that the datacenters do not collude.

Our framework has the following security and performance requirements. First, the datacenters must not infer any information from the stored SNPs, so both the states and positions of SNPs must be stored in encrypted form. To meet this requirement, we use Shamir's (l,n) secret sharing scheme to hide the SNP states and a symmetric encryption scheme to hide the SNP positions. We also assume that an adversary cannot access $l \leq n$ or more datacenters at any time. Our second security requirement is that the hospital must not know any information other than the result of the DST. To meet this requirement, we perform the DST operations at the datacenters and send the encrypted result to the hospital. Our final security requirement is that the datacenters must not infer any information about the nature of the DST from the data they receive from the hospital. We use our own obfuscation techniques to achieve this goal. As for performance concerns, we need to ensure that both the storage and computational overheads of our framework are low.

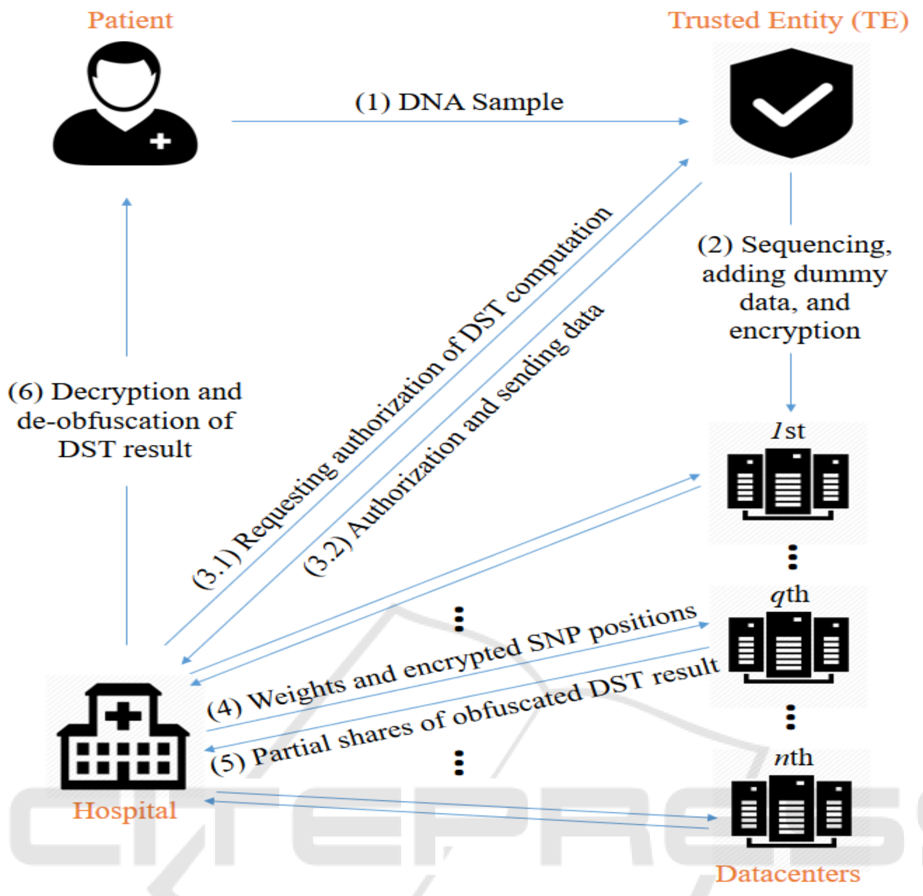
Patient
Trusted Entity (TE)
(1) DNA Sample
(2) Sequencing, adding dummy data, and encryption
(3.1) Requesting authorization of DST computation
(3.2) Authorization and sending data
(4) Weights and encrypted SNP positions
(5) Partial shares of obfuscated DST result
(6) Decryption and de-obfuscation of DST result
1st
qth
nth
Hospital
Datacenters

Figure 2: The architecture and workflow of our scheme.

4.2 Workflow

The working of our scheme is summarized in Figure 2 and explained below step-by-step.

Step 1 [Preliminary Step]. Our scheme begins with the patient sending her biological DNA sample to the TE for sequencing and future processing. The patient is involved in this step only.

Step 2 [Preprocessing and Distribution]. In this step, the TE first sequences the DNA sample, and then obtains the SNP states and positions in digital format. Next, the TE creates a number of *dummy SNPs* and mix them with the *actual SNPs* (SNPs obtained from sequencing). The introduced dummy SNPs will be used by the hospital to obfuscate the DST operations. These SNPs are finally encrypted by the TE and sent to the datacenters. Note that this step and the previous step (Step 1) are one-time operations. Also note that in the rest of the paper, the term SNPs refers to the mix of actual and dummy SNPs.

The TE encrypts a SNP by encrypting its state and position. The position is encrypted with the same symmetric encryption method as the one used by Ayday *et al.* (Ayday et al., 2013). We will therefore omit a detailed discussion on the encryption of positions in this paper, but note that in our scheme, the symmetric key used for position encryption is established between the TE and the hospital.

We use Shamir's secret sharing to encrypt SNP states. From Section 3, we know that the state of a SNP (which is the secret) is an integer from the set $\{0,1,2\}$. To encrypt a state, the TE chooses a prime number p, and using Equation 2, obtains the q^{th} (where $1 \leq q \leq n$) share as

$$F(\text{SNP}, q) = (\text{SNP} + \alpha_q) \bmod p.$$

The TE then sends $F(\text{SNP}, q)$ along with the encrypted SNP position to the q^{th} datacenter.

Step 3 [DST Authorization]. In this step, the hospital seeks DST authorization from the TE for a particular disease. To obtain authorization, the hospital requests from the TE the list of actual SNPs that is required to perform the DST. The TE then decides whether the hospital's request is legitimate, and if so, the TE sends the encrypted positions of the actual SNPs to the hospital. The TE also sends the states and positions of a number of dummy SNPs to the hospital. The exact number and type of dummy SNPs will be addressed in detail in Section 5.

Step 4 [Obfuscation and Computation Request]. In this step, the hospital obfuscates the DST operations to prevent datacenters from learning any information about the nature of the DST. Since a datacenter might be able to infer the type of the disease being tested for if it knows the number of actual SNPs used in the DST or the weights associated with these SNPs, the hospital cannot simply send the positions and plaintext weights of actual SNPs to the datacenters. Instead, the hospital produces a mixture of actual SNPs and dummy SNPs (obtained from TE in Step 3) and generates dummy weights for dummy SNPs. This mixture of actual and dummy SNPs, along with the actual and dummy weights, are then sent to the datacenters. For additional security, the hospital should also split the DST computation into multiple parts by simply dividing the SNP mixture into multiple parts. Each part of the DST computation can then be carried out independently with the hospital sending each corresponding part of the SNP mixture to the datacenters. In the end, the hospital will be able to aggregate the partial results and produce the desired disease susceptibility.

Step 5 [Encrypted Domain DST]. In this step, the datacenters first fetch the required SNP states (in the form of encrypted shares) from their database using the encrypted SNP positions they have received from the hospital. Using these shares of SNP states, *i.e.,* $F(\mathrm{SNP}_i, q)$'s, and the plaintext weights provided by the hospital, *i.e.,* C_i^X's and $p_j^i(X)$'s, the datacenters can perform the obfuscated DST computation. We use $L_X{}'$ to denote the set of dummy SNPs involved in the obfuscated DST computation, and L_X to denote the union of dummy and actual SNPs. In the end, each datacenter will obtain a share of the obfuscated DST result. We denote the obfuscated result $S_P^{X'}$ and the qth share of the result $F(S_P^{X'}, q)$.

One technical difficulty arises from the fact that SNP weights are floating point numbers, which are incompatible with the modular prime operations performed by Shamir's secret sharing. To overcome this issue, the datacenters must convert the floating point numbers to integers. Given a precision d, the datacenters will first round a float to d decimal places and then multiplying the roundoff value by 2×10^d. The multiplication by 2 is necessary here because the DST formula contains divisions by 2. For simplicity, we will also denote the integral SNP weights using C_i^X and $p_j^i(X)$ in this paper.

After converting SNP weights to integers, the datacenters will be ready to compute their respective shares of the obfuscated disease susceptibility and send the shares back to the hospital. The computation is done by using Equation 1 as:

$$F(S_P^{X'}, q) = \frac{1}{\sum_{i \in L} C_i^X} \times \sum_{i \in L} C_i^X \Big[c_0 \cdot F(\mathrm{SNP}_i, q)^2 + c_1 \cdot F(\mathrm{SNP}_i, q) + p_0^i(X) \Big],$$

where

$$c_0 = \frac{1}{2}[p_0^i(X) - 2p_1^i(X) + p_2^i(X)]$$

and

$$c_1 = \frac{1}{2}[-3p_0^i(X) + 4p_1^i(X) - p_2^i(X)].$$

By substituting $F(\mathrm{SNP}_i, q) = (\mathrm{SNP}_i + \alpha_q) \bmod p$ in the above equation, we get

$$F(S_P^{X'}, q) = (S_P^{X'} + \beta_q) \bmod p,$$

where

$$\beta_q = \frac{1}{\sum_{i \in L} C_i^X} \times \sum_{i \in L} C_i^X \Big[c_0 \cdot \alpha_q^2 + (2c_0 \times \mathrm{SNP}_i + c_1) \cdot \alpha_q \Big]$$

is a constant.

Note that since α_q is a value of an $(l-1)$-degree polynomial, β_q is a value of a $(2l-2)$-degree polynomial. Thus, $F(S_P^{X'}, q)$ is a share created from a $(2l-1)$-degree secret sharing polynomial:

$$F(x) = (S_P^{X'} + \beta_x) \bmod p.$$

Step 6 [Obtaining the DST Result]. In the final step of our scheme, the hospital obtains the final DST result (S_P^X) from the shares it received from the datacenters in Step 5. Since the hospital divided the DST computation into a number of smaller parts in Step 4, the hospital would have also received in Step 5 multiple partial shares from each datacenter corresponding to the parts. Therefore, to obtain the final DST result, the hospital first needs to reconstruct a complete share from the partial shares received from each datacenter. The hospital then recovers the obfuscated DST result in plaintext from the complete shares. Finally, the hospital de-obfuscate the obfuscated result by removing the noise introduced by dummy SNP and their weights, obtaining the final DST result.

Suppose the hospital split the DST computation into t parts. The SNPs included in the mth part are denoted L_m ($L_m \subset L$, and $\cup_{m=1}^{t} L_m = L$), and let

$F_m(S_P^{X\prime},q)$ be the partial share that the qth datacenter produced in the mth part of the DST computation. Using all t $F_m(S_P^{X\prime},q)$'s, the hospital can reconstruct the complete share $F(S_P^{X\prime},q)$ as follows:

$$F(S_P^{X\prime},q) = \frac{\prod_{m=1}^{t}\left[F_m(S_P^{X\prime},q)\cdot\sum_{i\in L_m} C_i^X\right]}{\sum_{m=1}^{t}\sum_{i\in L_m} C_i^X}.$$

After reconstructing complete shares from at least $(2l-1)$ datacenters, the hospital uses Lagrange interpolation to recover the obfuscated DST result, $S_P^{X\prime}$, in plaintext. Finally, the hospital de-obfuscates $S_P^{X\prime}$ and obtains the final DST result S_P^X using:

$$S_P^X = S_P^{X\prime}\cdot\frac{\sum_{i\in L_X} C_i^X}{\sum_{i\in L} C_i^X} - \frac{\sum_{i\in L'_X,\, j=\mathrm{SNP}_i} C_i^X\cdot P_j^i(X)}{\sum_{i\in L_X} C_i^X}.$$

5 ANALYSIS AND EXPERIMENT

In this section, we present the security assurance and performance overhead of our system. We also compare the our scheme with Ayday *et al.*'s scheme.

We implemented a prototype of our scheme using Java and simulated the scheme on a PC with Intel $i5-5300U$ and 12 GB of RAM running on Windows 10. Our prototype uses Shamir's $(3,6)$ secret sharing with 6 datacenters, and shares of at least 5 datacenters are required to recover the secret DST result. To convert floating point SNP states to integers, our prototype rounds them to 4 decimal places. Using the same technology, we also implemented and simulated Ayday *et al.*'s scheme with a 2048-bit key Paillier cryptosystem[1].

In our experiment, we used real patient SNP data obtained from (Personal-Genome-Project, 2016) and disease markers from (Eupedia, 2016). We used (dbSNP, 2016) to determine SNP states of the patient, and generated SNP weights by choosing a random floating point number in $[0,1)$. We validated the correctness of our scheme by comparing the result of our encrypted-domain DST with the result of the plaintext DST.

5.1 Security Analysis

We believe that the use of Shamir's (l,n) secret sharing makes our scheme more secure than Ayday *et al.*'s scheme. Our scheme does not require an encryption key and enables the hospital to verify the integrity of test results when the number of shares required to obtain the secret DST result (*i.e.,* $k=2l-1$) is less than the number of deployed datacenters (*i.e., n*). In such cases, the secret result can be obtained by the hospital in $\binom{n}{k}$ ways, and any inconsistency would indicate that at least one datacenter has been compromised.

Moreover, as shown by Barman *et al.*, malicious hospitals are able to infer SNP states by carefully crafting the SNP weights that are to be used in DST computations (Barman et al., 2015). Unlike Ayday *et al.*'s scheme, our scheme is secure against such attacks because plaintext SNP weights are exposed to the datacenters, and the datacenters can simply terminate the DST computation when they see suspicious weights (*e.g.,* 0's and consecutive powers). However, it is also important to note that the number and values of plaintext SNP weights are sensitive information that could be used to infer the type of the disease being tested for. Our scheme addresses this vulnerability through the use of dummy SNP and dummy weights, and this obfuscation technique merits a more detailed discussion.

Dummy SNPs and dummy weights have two functions. Their first function is to hide the number of actual SNPs involved in a particular DST. Given that no disease listed on (Eupedia, 2016) is linked to more than 100 SNPs, only a few dozen dummy SNPs are required for this purpose in each DST. Even if we take into account the need to choose a different set of dummy SNPs for different DSTs, the total number of dummy SNPs that need be stored at the datacenters is trivial compared to 4 million, the average number of SNPs in an individual's genome.

The second function of dummy SNPs and weights is to introduce noise to the values of actual SNP weights. Note that in order to effectively accomplish this goal, the hospital cannot simply randomly generate dummy SNP weights for every DST computation. Consider the following scenario. The hospital tests several patients for the same disease, and for each patient, the hospital sends, along with actual encrypted SNP positions and their plaintext weights, a number of dummy encrypted SNP positions and randomly generated dummy weights to the datacenters. Since the dummy weights are randomly generated, they will likely be different for different patients. However, the actual SNP weights will be exactly the same because the patients are all being tested for the same disease. Exploiting this discrepancy in patterns, the datacenters will likely be able to separate the actual SNPs from the dummy SNPs by identifying a set of SNP weights that are present in the DST for every patient.

To overcome this threat, our scheme requires the hospital to associate a set of fixed dummy weights with each disease in order to ensure that no only actual SNP weights, but also dummy SNP weights, are

[1]The code of the proposed method can be found at https://github.com/guyu96/encrypted-domain-DST

consistent across different patients. For added security, our scheme also requires that the hospital divide a single DST computation into multiple parts, as explained in Step 4 of our workflow, making it impossible for datacenters to determine the exact mixture of actual and dummy SNPs involved in any particular DST. Again, note that a few dozen dummy SNPs are sufficient for camouflaging the values of actual SNP weights as long as dummy weights are carefully and consistently chosen.

5.2 Data Overhead

We analyze the amount of storage required by both our scheme and Ayday *et al.*'s scheme in this section.

Since the number of dummy SNPs is smaller than the number of actual SNPs by several orders of magnitude, for our scheme, we only consider the data overhead incurred by the use of Shamir's secret sharing. For each 2-bit SNP state, we create n shares, each of which is a 64-bit integer. Each 2-bit SNP state therefore requires $64n$ bits of storage. In our implementation, $n = 6$ and as a result 192 times as much storage is required compared to the unencrypted-domain DST. However, our scheme significantly improves upon Ayday *et al.*'s scheme and requires approximately 40 times less space. To store the states of 4 million SNPs (average number of SNPs in an individual), our scheme requires 183 MB of storage whereas Ayday *et al.*'s scheme requires 7.63 GB of storage.

Communications between different parties during the DST computation are also much more efficient in our scheme. Suppose that z_1 actual SNPs and z_2 dummy SNPs are involved in the computation, and the hospital splits the computation into t parts. As shown in the security analysis, setting $z_2 = z_1$ is sufficiently secure and we denote this value z. Since our implementation rounds SNP weights to 4 decimal places, the integral weights can be represented by 16-bit integers.

In our scheme, three transmissions are involved in the DST computation: (1) the TE sends dummy SNP states and encrypted SNP positions (both actual and dummy) to the hospital (which requires $258z$ bits); (2) the hospital sends SNP weights and encrypted SNP positions to n datacenters in t parts ($288nz$ bits); and (3) each of the k datacenters sends t partial shares to the hospital ($64tk$ bits). Therefore, in our scheme $(258+288n)z+64tk$ bits are transmitted in total.

On the other hand, Ayday *et al.*'s scheme includes four transmissions: (1) the patient sends encrypted SNP positions to the datacenter ($128z$ bits); (2) the datacenter sends encrypted SNP states to the hospital ($8192z$ bits); (3) the hospital sends the encrypted DST result to the datacenter (8192 bits); and (4) the datacenter sends the partially decrypted result to the hospital (8192 bits). In total, $8320z+16384$ bits are transmitted in Ayday *et al.*'s scheme.

In our implementation, we have $n = 6$, $k = 5$, and $t = 8$. Thus the total amount of data transmitted in our scheme is $1986z+2560$ bits, which is more than 4 times less than the amount of transmission required by Ayday *et al.*'s scheme.

5.3 Computational Cost

We compare the computational costs of our scheme with that of Ayday *et al.*'s scheme in this section. Since the encryption and distribution of SNP data can be preprocessed, we are only concerned with the encrypted-domain DST computation.

Since we use Shamir's secret sharing, we need only additions, multiplications and divisions of 64-bit integers. Ayday *et al.*'s scheme, however, requires more expensive operations on larger numbers (*i.e.,* multiplications and modular exponentiations of 4096-bit integers). Therefore, our scheme incurs a significantly smaller computational cost than Ayday *et al.*'s scheme. We carried out DST computations on 21 SNPs related to Type-2 Diabetes, and on average, our scheme runs $10,000$ times faster than Ayday *et al.*'s scheme, taking less than 1 millisecond compared to Ayday *et al.*'s 8 seconds.

6 CONCLUSION

Human genome-based disease susceptibility test (DST) has serious privacy concerns, and previous attempts at making DST secure all have notable drawbacks, including high storage overhead, slow computation speed, reliance on patient involvement and even vulnerabilities to certain kinds of attacks. To address these issues, we propose in this paper a more practical privacy-preserving DST scheme that leverages the additive and pseudo-multiplicative homomorphism of Shamir's secret sharing. As demonstrated by both theoretical analysis and empirical evidence, our scheme is more secure and has significantly improved space and time efficiencies in comparison to the seminal work by Ayday *et al.*.

REFERENCES

Ayday, E., Raisaro, J. L., Hubaux, J.-P., and Rougemont, J. (2013). Protecting and evaluating genomic privacy in medical tests and personalized medicine. In *WPES*, pages 95–106.

Barman, L., Elgraini, M. T., Raisaro, J. L., Hubaux, J. P., and Ayday, E. (2015). Privacy threats and practical solutions for genetic risk tests. In *IEEE SPW*, pages 27–31.

Bresson, E., Catalano, D., and Pointcheval, D. (2003). A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In *ASIACRYPT*, volume 2894, pages 37–54.

Butts, C., KamelReid, S., Batist, G., and et al (2016). Benefits, issues, and recommendations for personalized medicine in oncology in canada. *Current Oncology*, 20(5):e475–e483.

Danezis, G. and De Cristofaro, E. (2014). Fast and private genomic testing for disease susceptibility. In *WPES*, pages 31–34.

dbSNP (2016). Home page. http://www.ncbi.nlm.nih.gov/projects/SNP/.

Djatmiko, M., Friedman, A., Boreli, R., Lawrence, F., Thorne, B., and Hardy, S. (2014). Secure evaluation protocol for personalized medicine. In *WPES*, pages 159–162.

Eupedia (2016). List of alleles (snp's) linked to physical and physcological traits, medical conditions and diseases. http://www.eupedia.com/genetics/medical_dna_test.s html.

Gennaro, R., Rabin, M. O., and Rabin, T. (1998). Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *ACM PODC*, pages 101–111, Puerto Vallarta, Mexico.

Huang, Z., Ayday, E., Fellay, J., Hubaux, J. P., and Juels, A. (2015). Genoguard: Protecting genomic data against brute-force attacks. In *IEEE S&P*, pages 447–462.

Jha, S., Kruger, L., and Shmatikov, V. (2008). Towards practical privacy for genomic computation. In *IEEE S&P*, pages 216–230.

Johnson, A. D. and O'Donnell, C. J. (2009). An open access database of genome-wide association results. *BMC Medical Genetics*, 10(1):1–17.

Karvelas, N., Peter, A., Katzenbeisser, S., Tews, E., and Hamacher, K. (2014). Privacy-preserving whole genome sequence processing through proxy-aided oram. In *WPES*, pages 1–10.

Kathiresan, S., Melander, O., Anevski, D., and et al. (2008). Polymorphisms associated with cholesterol and risk of cardiovascular events. *NEJM*, 358:1240–1249.

Mohanty, M., Atrey, P. K., and Ooi, W. T. (2012). Secure cloud-based medical data visualization. In *ACMMM*, pages 1105–1108, Nara, Japan.

Mohanty, M., Ooi, W. T., and Atrey, P. K. (2013a). Scale me, crop me, know me not: supporting scaling and cropping in secret image sharing. In *IEEE ICME*, San Jose, USA.

Mohanty, M., Ooi, W. T., and Atrey, P. K. (2013b). Secure cloud-based volume ray-casting. In *IEEE CloudCom*, Bristol, UK.

Naveed, M., Ayday, E., Clayton, E. W., and et al. (2015). Privacy in the genomic era. *ACM Computing Surveys*, 48(1):6:1–6:44.

Personal-Genome-Project (2016). Public genetic data. https://my.pgp-hms.org/public_genetic_data.

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