Privacy-Enhancing Technologies for Medical Tests Using Genomic Data

Technical Report

Erman Ayday, Jean Louis Raisaro and Jean-Pierre Hubaux
School of Computer and Communication Sciences
Ecole Polytechnique Federale de Lausanne (EPFL)
firstname.lastname@epfl.ch

Abstract*—In this paper, we propose privacy-enhancing technologies for medical tests and personalized medicine methods, which utilize patients' genomic data. First, we highlight the potential privacy threats on genomic data and the challenges of providing privacy-preserving algorithms. Then, focusing specifically on a typical disease-susceptibility test, we develop a new architecture (between the patient and the medical unit) and propose privacy-preserving algorithms by utilizing homomorphic encryption and proxy encryption. Assuming the whole genome sequencing is done by a certified institution, we propose to store patients' genomic data encrypted by their public keys at a Storage and Processing Unit (SPU). The proposed algorithm lets the SPU (or the medical unit) process the encrypted genomic data for medical tests and personalized medicine methods while preserving the privacy of patients' genomic data. We extensively analyze the relationship between the storage cost (of the genomic data), the level of genomic privacy (of the patient), and the characteristics of the genomic data. Furthermore, we implement and show via a complexity analysis the practicality of the proposed schemes. Finally, we evaluate the security of the proposed schemes and propose new research directions on genomic privacy.

I. Introduction

Privacy control can be defined as the ability of individuals to determine when, how, and to what extent information about themselves is revealed to others. In this way, the usage of private data will remain in context and it will be used exclusively for the purpose the data owner has in mind. Privacy is usually protected by both legal and technological means. By using legal actions, such as data protection directives and fair information practices, privacy regulations can enforce privacy protection on a large scale. Yet, this approach is mostly reactive, as it defines regulations after technologies are put in place. To avoid this issue, Privacy-Enhancing Technologies (PETs) [1]–[3] can be incorporated into the design of new systems in order to protect individuals' data. PETs protect privacy by eliminating or obfuscating personal data, thereby preventing misuse or involuntary loss of data, without affecting the functionality of the information system [4]. Their objective is to make it difficult for a malicious entity to link information to specific users. In order to obfuscate personal data, PETs often rely on cryptographic primitives, such as anonymous authentication and encryption.

Genomics is becoming the next significant challenge for privacy. The price of a complete genome profile has plummeted below $\$200$ for genome-wide genotyping (i.e., the characterization of about one million common genetic variants), which is offered by a number of companies (located mostly in the US). Whole genome sequencing is also offered through the same direct-to-consumer model (but at a higher price). This low cost of DNA sequencing will break the physician/patient connection, because private citizens (from anywhere in the world) can have their genome sequenced without involving their family doctor. This can open the door to all kinds of abuse, not yet fully understood.

As a result of the rapid evolution in genomic research, substantial progress is expected in terms of improved diagnosis and better preventive medicine. However, the impact on privacy is unprecedented, because (i) genetic diseases can be unveiled, (ii) the propensity to develop specific diseases (such as Alzheimer's) can be revealed, (iii) a volunteer accepting *de facto* to have his genomic code made public (as it already happened) can leak substantial information about his ethnic heritage and genomic data of his relatives (possibly against their will), and (iv) complex privacy issues can arise if DNA analysis is used for criminal investigations and insurance purposes. Such issues could lead to genetic discrimination (e.g., ancestry discrimination or discrimination due to geographic mapping of people). Even though the Genetic Information Nondiscrimination Act (GINA), which prohibits the use of genomic information in health insurance and employment, attempted to solve some of these problems in the US, these types of laws are very difficult to enforce.

An even more severe case, currently not widely considered, is where a malicious party initiates a cross-layer attack by utilizing privacy-sensitive information belonging to a victim retrieved from different sources (e.g., genomic data, location, online social network, etc.), thus creating the opportunity for a large variety of fraudulent uses of such data. For example, as stated in the Personal Genome Project (PGP) consent form [5], a malicious party could make synthetic DNA of a victim and plant it at a crime scene to falsely accuse him. In this hypothetical situation, the attacker can make his accusation stronger if he has the location patterns of the victim to be blamed, and hence, knows that the victim was close to the crime scene at the time of the crime. Similarly, an attacker can easily obtain information on close relatives of a target from online social network data, thus effectively increasing the potential access to target's genomic data if his relatives' DNA has been sequenced. In other words, even if the victim has perfect privacy on his own genome, if the attacker has access to the DNA sequence of the relatives, he can obtain significant information about the victim's DNA sequence.

Even though, at this stage, the field of genomics is generally free from serious attacks, it is likely that the above threats will become more serious as the number of sequenced individuals becomes larger. Such was the case of the Internet that was initially run and used by well-intentioned researchers. However, once it became more widely used, it became plagued by uncountable attacks such as spyware, viruses, spam, botnets, Denial-of-Service attacks, etc. Therefore, the need to adapt PETs to personal genomic data will only grow with time, as they are key tools for preventing an adversary from linking particular genomic data to a specific person or from inferring privacy-sensitive genomic data about a person.

It is obvious that users need to reveal personal and even privacy-sensitive information for genomic tests (e.g., paternity tests, disease-susceptibility tests, etc.). Nevertheless, they want to control how this information is used by the service providers (e.g., medical units such as healthcare centers or pharmaceutical companies, depending on the type of the test) [6]. Currently, the companies and hospitals that perform DNA sequencing store the genomic data of their customers and patients. Of course, tight legislation regulates their activities, but it is extremely difficult for them to protect themselves against the misdeeds of a hacker or a disgruntled employee. In a non-adversarial scenario, however, making use of this data requires legitimate professionals (e.g., physicians and pharmacists) to access the data in some way. Therefore, new architectures and protocols are needed to store and process this privacy-sensitive genomic data, while still enabling its utilization by the service providers (e.g., medical units).

In this work, our goal is to protect the privacy of users' genomic data while enabling medical units to access the genomic data in order to conduct medical tests or develop personalized medicine methods. In a medical test, a medical center checks for different health risks (e.g., disease susceptibilities) of a user by using specific parts of his genome. Similarly, to provide personalized medicine, a pharmaceutical company tests the compatibility of a user on a particular medicine, or a pharmacist checks the compatibility of a given medicine (e.g., over-the-counter drug) to a given user. In both scenarios, in order to preserve his privacy, the user does not want to reveal his complete genome to the medical center or to the pharmaceutical company. Moreover, in some scenarios, it is the pharmaceutical companies who do not want to reveal the genetic properties of their drugs. To achieve these goals, we propose to store the genomic data at a *Storage and Processing Unit* (SPU) and conduct the computations on genomic data utilizing homomorphic encryption and proxy encryption to preserve the privacy of the genomic data.

The rest of the paper is organized as follows. In the rest of this section, we discuss the challenges in genomic privacy and summarize the related work on genomic privacy. In Section II, we describe our proposed schemes for privacy-preserving medical tests and personalized medicine. Furthermore, we analyze the level of privacy provided by the proposed schemes for different design and genomic criteria. Then, in Section III, we discuss the implementation of the proposed schemes and present their complexity and security evaluations. Finally, in Section IV, we conclude the paper and discuss new research directions on genomic privacy.

A. Challenges of Genomic Privacy

Obviously, there are certain obstacles for achieving our goals on genomic privacy. These are mostly due to (i) the balance between privacy and reliability of the genomic data, (ii) the structure of the human genome, and (iii) the evolution of the genomic research.

PETs generally protect users' privacy by either breaking the link between individuals' identities and the data they provide (e.g., removing user's identities from the published genomic data), or by decreasing the information provided (e.g., by using cryptographic tools or obfuscation techniques). Both techniques might reduce the reliability and the accuracy of the genomic data. Thus, a major issue to be addressed when designing PETs is limiting private information leakage while keeping an acceptable level of reliability and accuracy of the genomic data for the researchers and medical units.

Moreover, developing PETs for genomic data has many unique challenges, due to the architecture of the human genome. 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). The human genome consists of approximately three billion letters. Existing privacy-preserving methods do not scale to these large genomic data sizes; hence current algorithms are inadequate for privacy protection on the genomic level.

Finally, the rapid evolution in the field of genomics produces many new discoveries every year, which cause significant changes in the known facts. For example, the sensitivity of certain genomic information will change over time; hence it is crucial to develop dynamic algorithms that can smoothly adapt to this rapid evolution.

B. Related Work

Due to the sensitivity of genomic data, research on the privacy of genomic data has considerably accelerated over the past few years. We can put the research on genomic privacy in three main categories: (i) private string searching and comparison, (ii) private release of aggregate data, and (iii) private clinical genomics.

In [7], Troncoso-Pastoriza *et al.* propose a protocol for string searching, which is then improved by Blanton and Aliasgari [8]. In this approach, one party with his own DNA snippet can verify the existence of a short template within his snippet by using a Finite State Machine in an oblivious manner. To compute the similarity of DNA sequences, in [9], Jha *et al.* propose techniques for privately computing the edit distance of two strings by using garbled circuits. In [10], Bruekers *et al.* propose privacy-enhanced comparison of DNA profiles for identity, paternity and ancestry tests using homomorphic encryption. Similar to our work, in [11], Kantarcioglu *et al.* propose using homomorphic encryption to perform scientific investigations on integrated genomic data. As opposed to [11], we focus on personal use of genomic data (e.g., in medical tests and personalized medicine methods). In one of the recent works [12], Baldi *et al.* make use of medical tools and private string comparison for privacy-preserving paternity tests, personalized medicine, and genetic compatibility tests. Instead of utilizing public key encryption protocols, in [13], Canim *et al.* propose securing the biomedical data using cryptographic

hardware. Finally, in [14], Eppstein *et al.* propose a privacy-enhanced method for comparing two compressed DNA sequences by using Invertible Bloom Filter [15].

When releasing databases consisting of aggregate genomic data (e.g., for research purposes), it is shown that known privacy-preserving approaches (e.g., de-identification) are ineffective on (un-encrypted) genomic data [16], [17]. Homer *et al.* [18] prove that the presence of an individual in a case group can be determined using aggregate allele frequencies and his DNA profile. In another recent study [19], Gitschier shows that a combination of information, from genealogical registries and a haplotype analysis of the Y chromosome collected for the HapMap project, allows for the prediction of the surnames of a number of individuals held in the HapMap database. Thus, releasing genomic data (even in aggregate form) is currently banned by many institutions due to this privacy risk. In [20], Zhou *et al.* study the privacy risks of releasing the aggregate genomic data. They propose a risk-scale system to classify aggregate data and a guide for the release of such data. Recently, using differential privacy was proposed by Fienberg *et al.* [21]; they aim to ensure that two aggregated databases, differing from each other by only one individual's data (e.g., DNA sequence), have indistinguishable statistical features.

Recently, in [22], utilizing a public cloud, Chen *et al.* propose a secure and efficient algorithm to align short DNA sequences to a reference (human) DNA sequence (i.e., read mapping). Finally, in [23], Wang *et al.* propose a privacy-protection framework for important classes of genomic computations (e.g., search for homologous genes), in which they partition a genomic computation, distributing sensitive data to the data provider and the public data to the data user.

In this work, we focus on medical tests (e.g., disease-susceptibility test) and personalized medicine methods by using users' genomic data while protecting user's genomic privacy. As a result of our extensive collaboration with geneticists, clinicians, and biologists, we conclude that DNA string comparison (in which the medical unit can only check if the patient carries a specific combination of variants or not) is insufficient in many medical tests (that use genomic data) and would not be enough to pave the way to personalized medicine. As it will become clearer in the next sections, specific variants must be considered individually for each genetic test. Thus, as opposed to the above private string search and comparison techniques, which focus on privately comparing the distance between the genomic sequences, we use the individual variants of the users to conduct genetic disease susceptibility tests and develop personalized medicine methods. We consider the individual contribution of each variant to a particular disease, for which a string comparison algorithm (such as Private Set Intersection [24], [25]) would not work. Further, in our proposed algorithms, we consider the statistical relationship between the variants for the genomic privacy of the users. In addition, we make use of a Storage and Processing Unit (SPU) between the user (patient) and the medical unit to store the genomic data in encrypted form and make computations on it using homomorphic encryption and proxy encryption.

II. PETs for Medical Tests and Personalized Medicine Methods

In this work, we study the privacy issues of medical tests and personalized medicine methods. Most medical tests and personalized medicine methods (that use genomic data) involve a patient and a medical unit. In general, the medical unit is the family doctor, a physician, a pharmacist, a medical council, or an online service. In this study, we consider a malicious medical unit as the potential attacker. That is, a medical unit can be a malicious institution trying to obtain private genomic information about a patient (for which it is not authorized). Even if the medical unit is non-malicious, it is extremely difficult for medical units to protect themselves against the misdeeds of a hacker or a disgruntled employee, and hence, the attacker can also be considered as a hacker or a careless employee in the medical unit. Similarly, the genomic data is too sensitive to be stored on users' personal devices (mostly due to security, availability, and storage issues), hence it is risky to leave the users' genomic data in their own hands. In addition, extreme precaution is needed between the patient and the medical unit due to the sensitivity of genomic data. Thus, we believe that a Storage and Processing Unit (SPU) should be used to store and process the genomic data. We note that a private company (e.g., cloud storage service), the government, or a non-profit organization could play the role of the SPU. We also assume that the SPU is an honest organization, but it might be curious (e.g., existence of a curious party at the SPU), hence genomic data should be stored at the SPU in encrypted form (i.e., the SPU should not be able to access the content of patients' genomic data). This general architecture is illustrated in Fig. 1.

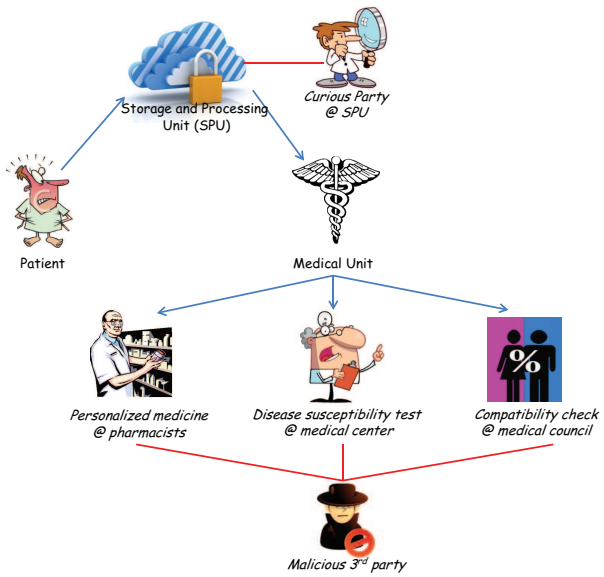


Fig. 1. General architecture between the patient, SPU, and the medical unit.

For the simplicity of presentation, in the rest of this section, we will focus on a particular medical test (namely, computing genetic disease susceptibility). We note that similar techniques would apply for other medical tests and personalized medicine methods. In a typical disease-susceptibility test, a medical center (MC) wants to check the susceptibility of a patient (P)

to a particular disease X (i.e., probability that the patient P will develop disease X). It is shown that a genetic disease-susceptibility test can be realized by analyzing particular Single Nucleotide Polymorphisms (SNPs) of the patient via some operations, such as weighted averaging [26] or Likelihood Ratio (LR) test [27]. A SNP is a position in the genome holding a nucleotide (A, T, C or G), which varies between individuals. For example, it is reported that there are three particular genes bearing a total of ten particular SNPs necessary to analyze a patient's susceptibility to Alzheimer's disease [28]. Each SNP contributes to the susceptibility in a different amount and the contribution amount of each SNP is determined by previous studies on case and control groups (these studies are published in several papers). Furthermore, some of the SNPs contribute to the development of a disease, whereas some are protective.

In general, there are two alleles (nucleotides which reside at a SNP position) observed at a given SNP position: (i) The major allele is the most frequently observed nucleotide, and (ii) the minor allele is the rare nucleotide. Everyone inherits one allele of every SNP location from each of his parents. If an individual receives the same allele from both parents, he is said to have a *homozygous* variant for that SNP location. If, however, he inherits a different allele from each parent (one minor and one major), he has a *heterozygous* variant. There are approximately 40 million approved SNPs in the human population as of now (according to the NCBI dbSNP [29]) and each patient carries on average 4 million SNPs (e.g., variants) out of this 40 million. Moreover, this set of 4 million SNPs is different for each patient. From now on, to avoid confusion, for each patient, we refer to these 4 million variants as the *real SNPs* and the remaining non-variants (approved SNPs that do not exist for the considered patient) as the *potential SNPs* of the patient; when we only say "SNPs", we mean both the real and potential SNPs.

At this point, it can be argued that these 4 million real SNPs (nucleotides) could be easily stored on the patient's computer or mobile device, instead of the SPU. However, we assert that this should be avoided due to the following issues. On one hand, the number of approved SNPs in human population continues to increase (even faster than the Moor's Law) with new discoveries. Further, types of variations in human population are not limited to SNPs, and there are other types of variations such as Copy-Number Variations (CNVs), rearrangements, or translocations[1], consequently the required storage per patient is likely to be considerably more than only 4 million nucleotides. This higher storage cost might still be affordable to an average patient (via desktop computers or USB drives), however, genomic data of the patient should be available any time (e.g., for emergencies), thus it should be stored at a reliable source such as the SPU. On the other hand, as we discussed before, leaving the patient's genomic data in his own hands and letting him store it on his computer or mobile device is risky, because his mobile device can be stolen or his computer can be hacked.

A potential attacker can learn about the susceptibilities of the patient to privacy-sensitive diseases if he obtains some specific real SNPs of the patient. Moreover, the knowledge of 75 real SNPs (out of approximately 4 million), if not fewer, will enable the attacker to identify a person [30]. These situations could lead to genetic discrimination such as denying a person's access to health (or life) insurance or obstructing his employment opportunities. As we discussed before, in our setting, both the MC and SPU pose a threat to the patient's privacy. On one hand, the MC can either be a malicious institution trying to obtain private information about the patient or it can be hacked by another malicious entity. On the other hand, the SPU is considered as an honest but curious entity. Thus, our goal is to build mechanisms in which the patient can preserve the privacy of his genomic sequence (his real SNPs) while enabling the MC to access his genomic data and conduct genetic tests.

We assume that the whole genome sequencing is done by a Certified Institution (CI) with the consent of the patient. Moreover, the genomic data of the patient is encrypted by the same CI (using the patient's public key) and uploaded to the SPU so that only the patient can decrypt the stored (potential or real) SNPs, and the SPU cannot access the SNPs of the patient. We are aware that the number of discovered SNPs increases with time. Thus, the patient's complete DNA sequence is also encrypted as a single vector file (via symmetric encryption using the patient's key) and stored at the SPU, thus when new SNPs are discovered, these can be included in the pool of the previously stored SNPs of the patient. We also assume the SPU does not have access to the real identities of the patients and data is stored at the SPU by using pseudonyms; this way, the SPU cannot associate the conducted genomic tests to the real identities of the patients. As an alternative, the privacy of the genomic data at the SPU can be further increased using *privacy enhanced access control* [31] or *Oblivious RAM (O-RAM) storage* [32] techniques, in which the data access patterns are completely hidden from the server (SPU).[2]

Depending on the access rights of the MC, the SPU can either (i) compute $\Pr(X)$, the probability that the patient will develop the disease X by checking the patient's encrypted SNPs via homomorphic encryption techniques [34][3], or (ii) provide the relevant SNPs to the MC (e.g., for complex diseases that cannot be interpreted using homomorphic operations). These access rights are defined either jointly by the MC and the patient or by the medical authorities. Further, access rights can be enforced by using a secure attribute-based system as in [35]. We note that homomorphic encryption lets the SPU (or MC) compute $\Pr(X)$ using encrypted SNPs of the patient P. In other words, the SPU (or MC) does not access P's SNPs to compute his predicted disease susceptibility. We use a modification of the Paillier cryptosystem (described in Section II-A) to support the homomorphic operations at the SPU (or MC).

We propose three different techniques for the storage and process of the SNPs at the SPU and the preservation of the patient's privacy: (i) Method 1 in Section II-B, (ii) Method 2 in Section II-C, and (iii) Method 3 in Section II-D. We describe these proposed techniques in detail in the following subsections. We also discuss the computation of genetic disease susceptibility by using homomorphic operations in

[1]Our proposed privacy-preserving mechanisms can be smoothly adapted for these alternative variations.

[2]Note that even the most efficient implementation of O-RAM introduces high storage overhead to the client (patient) [33], and it introduces $20 \sim 25$ times more overhead with respect to non-oblivious storage. Thus once it becomes more efficient, O-RAM storage could be considered as a future add-on to the proposed privacy-preserving mechanisms.

[3]In one of our proposed schemes (Method 3 in Section II-D), $\Pr(X)$ is computed at the MC via homomorphic operations.

General Notations	
SNP_i^P	Type of SNP i, SNP_i, of the patient P. $\mathrm{SNP}_i^P \in \{0,1\}$, 0 representing a potential SNP (i.e., non-variant) for P, and 1 representing a real SNP (i.e., a variant) for P.
$\mathbb{S}_P^X$	Predicted susceptibility of the patient P to disease X.
Υ_P	Set of real SNPs of the patient P (SNPs at which P has a variant: around 4 million at each patient).
Ω_P	Set of potential SNPs of the patient P (SNPs at which P does not have a variant: around 36 million at each patient).
Cryptographic Notations	
n, g	Public parameters of modified Paillier cryptosystem.
x	Weak secret key of the patient P.
$x^{(i)}$	i^{th} share of the patient P's secret key.
g^x	Public key of the patient P.
$\mathrm{E}(m, g^x)$	Encryption of message m with the patient P's public key.
Susceptibility Test via Weighted Averaging	
$\mathrm{p}_j^i(X)$	Probability that P would develop disease X, given $\mathrm{SNP}_i^P = j$, $\Pr(X \mid \mathrm{SNP}_i^P = j)$.
C_i^X	Contribution of SNP_i to the susceptibility to disease X.
Susceptibility Test via Likelihood Ratios	
I_X^P	Initial risk of the patient P for disease X.
$L_X^i(j)$	Likelihood Ratio (LR) when $\mathrm{SNP}_i = j$ for disease X.

TABLE I
NOTATIONS AND DEFINITIONS.

Section II-E. In the rest of this work, for simplicity of the presentation, we do not consider the type of the variant at a real SNP location (i.e., whether the variation is homozygous or heterozygous for that real SNP); we only consider whether the patient has a real SNP or not at a particular location. However, the proposed approaches and the analysis (in Section II-C) can easily be extended to cover the types of the variants. In order to facilitate future references, frequently used notations are listed in Table I for the different stages of the proposed schemes.

A. Paillier Cryptosystem

In this section, we briefly review the modified Paillier cryptosystem (described in detail in [34], [36]), which we use in this work, and its homomorphic properties.

The public key of the patient P is represented as $(n, g, h = g^x)$, where the strong secret key is the factorization of $n = pq$ (p, q are safe primes), the weak secret key is $x \in [1, n^2/2]$, and g of order $(p-1)(q-1)/2$. Such a g can be easily found by selecting a random $a \in \mathbb{Z}_{n^2}^*$ and computing $g = -a^{2n}$.

Encryption of a message: To encrypt a message $m \in \mathbb{Z}_n$, we first select a random $r \in [1, n/4]$ and generate the ciphertext pair (T_1, T_2) as below:

$$T_1 = g^r \mod n^2 \quad \text{and} \quad T_2 = h^r(1+mn) \mod n^2. \quad (1)$$

Re-encryption of a message: An encrypted message (T_1, T_2) can be re-encrypted under the same public key, using a new random number $r_1 \in [1, n/4]$ as below:

$$\hat{T_1} = g^{r_1} T_1 \mod n^2 \quad \text{and} \quad \hat{T_2} = h^{r_1} T_2 \mod n^2. \quad (2)$$

Decryption of a message: The message m can be recovered as follows:

$$m = \Lambda(T_2/T_1^x), \quad (3)$$

where $\Lambda(u) = \frac{(u-1) \mod n^2}{n}$, for all $u \in \{u < n^2 \mid u = 1 \mod n\}$.

Homomorphic properties: Assume two messages m_1 and m_2 are encrypted using two different random numbers r_1 and r_2, under the same public key, $(n, g, h = g^x)$, such that $\mathrm{E}(m_1, r_1, g^x) = (T_1^1, T_2^1)$ and $\mathrm{E}(m_2, r_2, g^x) = (T_1^2, T_2^2)$. Assume also that c is a constant number. Then the below-mentioned homomorphic properties are supported by Paillier cryptosystem:

• The product of two ciphertexts will decrypt to the sum of their corresponding plaintexts.

$$\mathrm{D}(\mathrm{E}(m_1, r_1, g^x) \cdot \mathrm{E}(m_2, r_2, g^x)) = \mathrm{D}(T_1^1 \cdot T_1^2, T_2^1 \cdot T_2^2 \mod n^2) = m_1 + m_2 \mod n. \quad (4)$$

• An encrypted plaintext raised to a constant c will decrypt to the product of the plaintext and the constant.

$$\mathrm{D}(\mathrm{E}(m_1, r_1, g^x)^c) = \mathrm{D}((T_1^1)^c, (T_2^1)^c \mod n^2) = cm_1 \mod n. \quad (5)$$

These homomorphic operations are conducted at the SPU (or MC depending on which approach is used) to compute the predicted susceptibility of the patient P to disease X, as will be discussed in Section II-E.

Proxy encryption: The patient's weak secret key x is randomly divided into two shares: $x^{(1)}$ and $x^{(2)}$ (such that $x = x^{(1)} + x^{(2)}$). $x^{(1)}$ is given to the SPU and $x^{(2)}$ is given to the MC. Using the above Paillier cryptosystem, an encrypted message (T_1, T_2) (under the patient's public key) can be partially decrypted by the SPU (using $x^{(1)}$) to generate the ciphertext pair $(\tilde{T_1}, \tilde{T_2})$ as below:

$$\tilde{T_1} = T_1 \quad \text{and} \quad \tilde{T_2} = T_2/T_1^{x^{(1)}} \mod n^2. \quad (6)$$

Now, $(\tilde{T}_1, \tilde{T}_2)$ can be decrypted at the MC using $x^{(2)}$ to recover the original message. $x^{(2)}$ can be provided to the MC once the patient is registered to the medical center or through the patient's digital ID card. Further details about the distribution of shares are out of the scope of this paper. We note that this approach is not proxy re-encryption; it is based on secret-sharing.

Overall, this modified Paillier cryptosystem is not key optimal, because the size of the MC's and SPU's secret storages do not remain constant. That is, both the MC and SPU need to store a secret for every patient. However, this storage cost can be considered negligible when compared to the storage of the genomic data. Further, the shares (e.g., $x^{(1)}$ and $x^{(2)}$) can be stored by the patient and sent to the MC and SPU only when it is needed in order to resolve this storage issue at the expense of extra communication overhead. Furthermore, the above modified Paillier cryptosystem is not proxy invisible, because all participants of the systems (i.e., P, MC and SPU) should be aware of the existence of the proxy. We discuss the security evaluation of this cryptosystem in Section III-B.

B. Method 1: Plaintext Locations at the SPU

In this approach, even though the SNPs of the patient are stored encrypted (via the patient's public key), the locations of the corresponding SNPs are stored in plaintext at the SPU. This is because, when a particular SNP (or set of SNPs) are queried by the MC, the SPU should know which SNPs to process (or send to the MC).

We assume that SNP_i at the patient P is represented as SNP_i^P and $\text{SNP}_i^P = 1$, if P has a real SNP (i.e., variant) at this location, and $\text{SNP}_i^P = 0$, if P does not have a variant at this location. We let Υ_P be the set of real SNPs of the patient P (at which $\text{SNP}_i^P = 1$). We also let Ω_P represent the set of potential SNPs (at which $\text{SNP}_i^P = 0$). As the locations of the SNPs are stored in plaintext, if the SPU only stores the real SNPs in Υ_P, a curious party at the SPU can learn all real SNP locations of the patient, and hence, much about his genomic sequence.[4] Therefore, the SPU stores the contents of both real and potential SNP locations (in $\{\Upsilon_P \cup \Omega_P\}$) in order to preserve the privacy of the patient. Below, we summarize the proposed approach for the privacy protecting disease-susceptibility test by using this particular storage technique. This approach is illustrated in Fig. 2.

• **Step 0:** The Cryptographic keys (public and secret keys) of each patient are generated and distributed to the patients during the initialization period. Then, symmetric keys are established between the parties, using which the communication between the parties is protected from an eavesdropper. We note that the distribution, update and revocation of cryptographic keys are handled by a trusted entity (similar to e-banking platforms).

• **Step 1:** The patient (P) provides his sample (e.g., his saliva) to the Certified Institution (CI) for sequencing.

• **Step 2:** The CI sequences P, and encrypts the contents of his real and potential SNP locations (in $\{\Upsilon_P \cup \Omega_P\}$) by using P's public key.

[4]The nucleotides corresponding to variants at particular locations of the DNA sequence are public knowledge. Thus, even though the contents of patient's real SNPs are encrypted, a curious party at the SPU can infer the nucleotides corresponding to these SNPs from their plaintext locations.

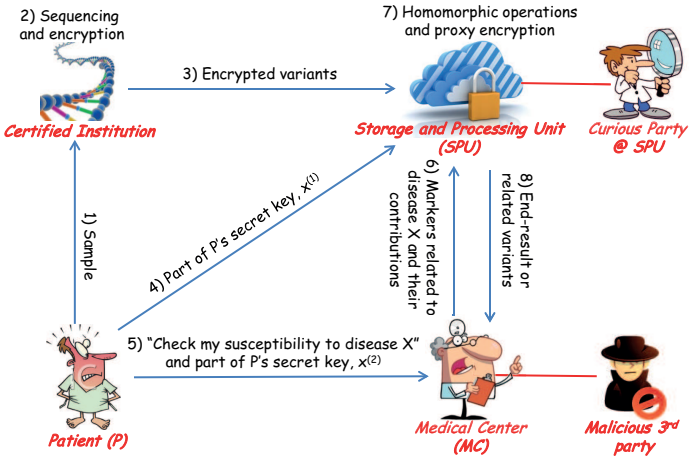


Fig. 2. Privacy-preserving protocol for disease-susceptibility test using Method 1 or Method 2.

• **Step 3:** The CI sends the encrypted SNPs of P to the SPU (so that the SPU cannot access to P's SNPs).

• **Step 4:** The patient provides a part of his secret key ($x^{(1)}$) to the SPU.

• **Step 5:** The MC wants to conduct a susceptibility test on P to a particular disease X, and P provides the other part of his secret key ($x^{(2)}$) to the MC.

• **Step 6:** The MC provides genetic variant markers, along with their individual contributions (to the disease susceptibility), to the SPU.

• **Step 7:** If the disease susceptibility can be interpreted by homomorphic operations, the SPU computes P's total susceptibility to disease X from the individual effects of SNPs by using the homomorphic properties of the Paillier cryptosystem as described in Section II-E. Otherwise, the SPU provides the relevant SNPs to the MC based on MC's access rights.

• **Step 7:** The SPU partially decrypts the end-result (or the relevant SNPs) using a part of P's secret key following a proxy encryption protocol (Section II-A).

• **Step 8:** The SPU sends the partially decrypted end-result (or the relevant SNPs) to the MC.

• **Step 9:** The MC decrypts the message received from the SPU using the other part of P's secret key and recovers the end-result (or the relevant SNPs).

The above technique provides the highest level of privacy and practicality for the patient, because (i) from the view point of a curious party at the SPU, inferring the locations of the patient's real SNPs with the stored information is equivalent to inferring them with no information about the patient, and (ii) the patient is not involved in the protocol after the sequencing (except for the consent between the patient and the MC for a particular test). However, this level of privacy and practicality comes at the cost of extra storage overhead at the SPU (due to the storage of both real and potential SNPs as discussed in Section III-A).

C. Method 2: Redundant Storage at the SPU

Due to the significant storage overhead (which is projected to increase with new discoveries in the field of genomics)

mentioned in Section II-B, here we propose another technique that reduces the storage overhead at the SPU at the expense of decrease in privacy. In a nutshell, we leave everything the same as in Section II-B, but, instead of storing the contents of all potential and real SNP locations, we store the real SNPs of the patient along with a certain level of redundancy (i.e., contents of some potential SNP locations). In other words, to mislead a curious party at the SPU, among the 40 million discovered SNPs, we store the approximately 4 million real SNPs (for which $\text{SNP}_i^P = 1, i \in \Upsilon_P$) along with some redundant content from Ω_P (with $\text{SNP}_j^P = 0$), for each patient.

Again, we assume that the location of the encrypted (real or potential) SNPs are stored in plaintext at the SPU and there exists a potential curious party at the SPU trying to infer the real SNPs of the patient (in Υ_P). An important issue to consider in this approach is the *Linkage Disequilibrium* (LD) between SNPs [37]. LD occurs when SNPs at the two loci (SNP positions) are not independent of each other. For simplicity, we represent the LD relationship between two SNPs i and j as $\Pr(\text{SNP}_i|\text{SNP}_j)$, where SNP_i (or SNP_j) takes values from the set $\{0, 1\}$.[5] We note that LD relationships are defined among all 40 million discovered SNPs, regardless of their type (i.e., real or potential) at a particular patient.

As in Section II-B, the SPU provides the end-result of a disease-susceptibility test or the relevant SNPs to the MC. However, in this case, if a particular potential SNP (requested by the MC or needed in the susceptibility test) is not stored at the SPU (i.e., $\text{SNP}_j^P = 0$), one of the following two scenarios occurs: (i) If the SPU provides the relevant SNPs to the MC, MC infers the missing potential SNPs from the reference genome (since it is known that the missing potential SNPs are not a variant for P), or (ii) if the SPU provides the end-result of the susceptibility test, the SPU uses the fact that $\text{SNP}_j^P = 0$ for each missing potential SNP j.

As expected, the amount of storage redundancy (due to the storage of the content from Ω_P), along with the LD between the SNPs and their characteristics, determine the level of a patient's genomic privacy. Therefore, in the rest of this section, we analyze the relationship between the amount of redundancy, LD values, characteristics of the SNPs, and the level of privacy. To do so, first, we observe the average probability of correctly inferring the locations of P's real SNPs (in Υ_P) considering varying amounts of redundancy and the LD values between the SNPs. That is, how much information from a patient's un-stored potential SNPs is revealed to the curious party at the SPU about the locations of his real SNPs? This problem can also be formulated similarly if the goal of the attacker is to determine the type of the variant at a real SNP location (e.g., homozygous or heterozygous).[6] It is worth noting that for this study, we create realistic models for the LD values and the characteristics of the SNPs. Further, for the created models, we try a wide range of parameters and observe a wide range of results to address most potential scenarios. However, as the field of genomics becomes more mature, our models can be replaced by the values obtained from the medical research.

[5]In compliance with genetic observations, we assume that the LD between two SNPs are not symmetric, i.e., $\Pr(\text{SNP}_i|\text{SNP}_j) \neq \Pr(\text{SNP}_j|\text{SNP}_i)$.

[6]In this case, SNP_i^P can take three different values from the set $\{0, 1, 2\}$, 0 representing a potential SNP (i.e., non-variant), 1 representing a real homozygous SNP, and 2 representing a real heterozygous SNP for P.

We let Ω_P^s and Ω_P^u denote the set of P's potential SNPs that are stored (for redundancy) and not stored at the SPU, respectively ($\Omega_P^s \cup \Omega_P^u = \Omega_P$). Further, K_i is the set of SNPs with which a particular SNP i has LD, and $|K_i| = k$ (for each SNP, these k SNPs are chosen among approximately 40 million SNPs). We assume that $k \geq 0$ and it is a truncated Gaussian random variable with only discrete values and obtained from a distribution with mean $\mu(k)$ and standard deviation $\sigma(k)$.

Initially, we compute $\Pr(\text{SNP}_i^P)$ for all (real and potential) SNPs in $\{\Upsilon_P \cup \Omega_P^s\}$ by using the LD relationships between these SNPs and those in Ω_P^u. As all SNPs in $\{\Upsilon_P \cup \Omega_P^s\}$ are encrypted and stored at the SPU, only the LD relationships between these SNPs and the un-stored SNPs in Ω_P^u are helpful for the curious party. Therefore, for each real SNP $i \in \Upsilon_P$, we observe $\Pr(\text{SNP}_i^P = 1|\text{SNP}_m^P = 0)$ for all $m \in \{K_i \cap \Omega_P^u\}$, get the average of these values, and compute $\Pr(\text{SNP}_i^P = 1)$. Similarly, for each potential SNP $j \in \Omega_P^s$, we observe $\Pr(\text{SNP}_j^P = 0|\text{SNP}_m^P = 0)$ for all $m \in \{K_j \cap \Omega_P^u\}$, average these values, and compute $\Pr(\text{SNP}_j^P = 0)$. We let l be the indicator of the LD strength between two SNPs. Thus, we represent $\Pr(\text{SNP}_i^P = 1|\text{SNP}_m^P = 0) = l$ ($i \in \Upsilon_P$, $m \in \{K_i \cap \Omega_P^u\}$) and $\Pr(\text{SNP}_j^P = 0|\text{SNP}_m^P = 0) = l$ ($j \in \Omega_P^s$, $m \in \{K_j \cap \Omega_P^u\}$) as truncated Gaussian random variables with range $[0.5, 1]$, obtained from a distribution with mean $\mu(l)$ and standard deviation $\sigma(l)$. Finally, if $|K_i| = k = 0$ or $|K_i \cap \Omega_P^u| = 0$ for a SNP i in $\{\Upsilon_P \cup \Omega_P^s\}$, we update $\Pr(\text{SNP}_i^P = 1)$ considering the fact that the expected value of all stored SNPs is known by the curious party as below:

$$\frac{1}{|\Upsilon_P \cup \Omega_P^s|} \sum_{j \in \Upsilon_P \cup \Omega_P^s} (SNP_j^P) \times \Pr(\text{SNP}_j^P) = \frac{|\Upsilon_P|}{|\Upsilon_P \cup \Omega_P^s|}. \tag{7}$$

In the following, we illustrate our numerical results that represent the relationship between storage, inference power of the curious party at the SPU, and LD values. We assume $|\Upsilon_P| = 4$ million and $|\Upsilon_P \cup \Omega_P| = 40$ million. We define the percentage of storage redundancy at the SPU as $\frac{|\Omega_P^s|}{|\Upsilon_P|} \times 100$ and compute the average value of $\Pr(\text{SNP}_i^P = 1)$ for a SNP in Υ_P for varying values of $\mu(k)$, $\sigma(k)$, $\mu(l)$, and $\sigma(l)$.[7] We repeat each simulation 100 times to obtain an average. Note that Method 1 (in Section II-B) is a special case of Method 2 (when the storage redundancy at the SPU is 900%), hence its privacy is the same as 900% redundancy in the following results.

In Fig. 3, we illustrate the variance in the average value of $\Pr(\text{SNP}_i^P = 1)$ for different values of $\mu(k)$, when $\mu(l) = 0.8$, $\sigma(l) = 0.15$, and $\sigma(k) = 0.75$. We note that "no LD" curve in the figure represents the case in which the LD values between the SNPs are ignored. We observe that as the available side information (i.e., number of un-stored potential SNPs in Ω_P^u having LD with the stored ones) increases, the inference power of the curious party increases, especially for low values of storage redundancy. For example, to have the same inference power for the curious party, 200% storage redundancy is required when $\mu(k) = 0$, whereas it is 700% when $\mu(k) = 4$. Furthermore, even at the maximum (i.e, 900%) storage redundancy, the curious party still has a slight

[7]Higher values of $\Pr(\text{SNP}_i^P = 1)$ indicate a higher inference power for the curious party at the SPU.

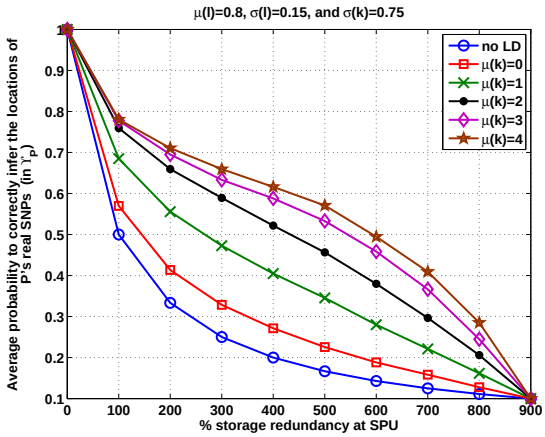


Fig. 3. Average probability to correctly infer the locations of patient's real SNPs (for the curious party at the SPU) with varying mean values of the number of LD pairs per SNP (i.e., $\mu(k)$) and storage redundancy.

probability of inferring the variants of the patient, because it knows that 4 out of 40 million of the stored content are variants. Next, in Fig. 4, we illustrate the variance in the same probability, this time for different values of $\mu(l)$, when $\mu(k) = 2$, $\sigma(k) = 0.75$, and $\sigma(l) = 0.25$.[8] As expected,

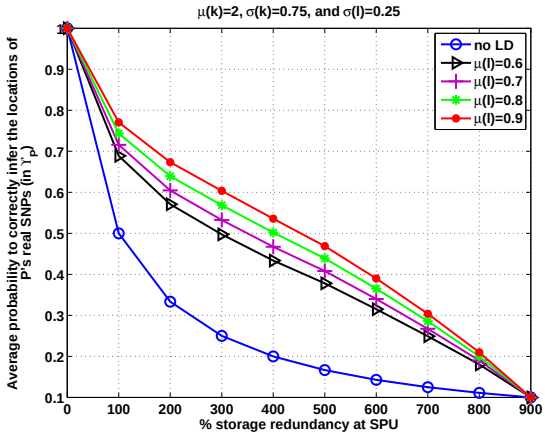


Fig. 4. Average probability to correctly infer the locations of patient's real SNPs (for the curious party at the SPU) with varying mean values of the LD strength between two SNPs (i.e., $\mu(l)$) and storage redundancy.

the inference power of the curious party increases when the strength of LD between the SNPs increases (i.e., when $\mu(l)$ increases). We observe that the strength of LD, however, does not affect the inference power as strong as k. Then, in Figs. 5 and 6, we show the $\text{Average}\{\Pr(\text{SNP}_i^P = 1)\}$ for varying standard deviations of k and l, and with 500% storage redundancy as follows: (i) in Fig. 5, we vary $\sigma(k)$ and $\mu(k)$, when $\mu(l) = 0.8$ and $\sigma(l) = 0.15$, and (ii) in Fig. 6, we vary $\sigma(l)$ and $\mu(l)$, when $\mu(k) = 2$ and $\sigma(k) = 0.75$. We observe that the inference power of the curious party varies (either increases or decreases) with an increasing value of $\sigma(k)$ ($\sigma(l)$) depending on $\mu(k)$ ($\mu(l)$), and, as expected, all curves converge to a single value for higher values of $\sigma(k)$ ($\sigma(l)$).

[8]For higher values of $\sigma(l)$, the gap between the different $\mu(l)$ curves becomes negligible, because the distribution becomes almost uniform, rather than truncated Gaussian.

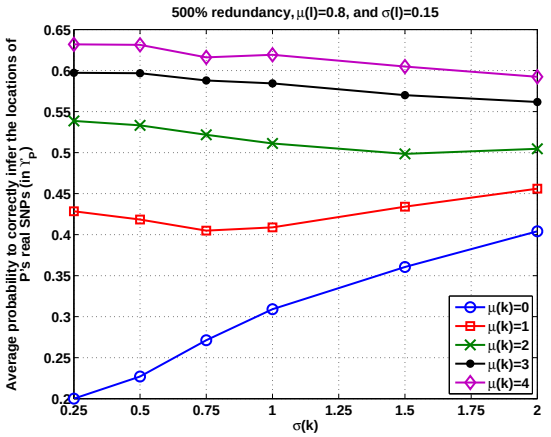


Fig. 5. Average probability to correctly infer the locations of patient's real SNPs (for the curious party at the SPU) with varying standard deviation and mean values of the number of LD pairs per SNP (i.e., $\sigma(k)$ and $\mu(k)$) and storage redundancy.

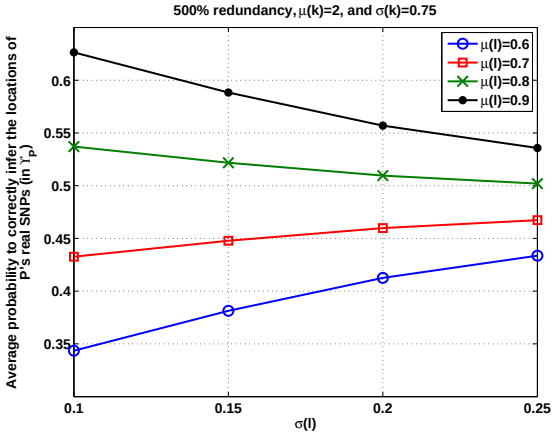


Fig. 6. Average probability to correctly infer the locations of patient's real SNPs (for the curious party at the SPU) with varying standard deviation and mean values of the LD strength between two SNPs (i.e., $\sigma(l)$ and $\mu(l)$) and storage redundancy.

Next, considering the individual characteristics of the real SNPs (i.e., their severity levels), we study the level of genomic privacy of a patient against a curious party at the SPU. The severity of a SNP i can be defined as the privacy-sensitivity of the SNP when $\text{SNP}_i^P = 1$ (i.e., when it exists as a variant at the patient P). For example, a real SNP revealing the predisposition of a patient for Alzheimer's disease can be considered more severe than another real SNP revealing his predisposition to a more benign disease. Severity values of the SNPs are determined as a result of medical studies (depending on their contributions to various diseases) and tables of disease severities provided by insurance companies (e.g., percentage of invalidity). We denote the severity of a real SNP i as V_i, and $0 \leq V_i \leq 1$ (1 denotes the highest severity). Thus, we define the genomic privacy of the patient P as below:

$$\Phi_P = -\sum_{i \in \Upsilon_P} \log_2 \left(\Pr(\text{SNP}_i^P = 1)\right) \times V_i. \qquad (8)$$

We do not use the traditional entropy metric [38], [39] to quantify privacy, as only one state of SNP_i^P poses privacy risks (i.e., $\text{SNP}_i^P = 1$), as discussed before.

First, we study the relationship between the storage redundancy and the severity of the real SNPs by focusing on three types of patients: (i) patient A, carrying mostly low severity real SNPs (in Υ_A), (ii) patient B, carrying mostly high severity real SNPs (in Υ_B), and (iii) patient C, carrying mixed severity real SNPs (in Υ_C). For each patient, the highest level of privacy is achieved when the storage redundancy is maximum (as in Method 1 in Section II-B). Thus, we recognize this level as 100% genomic privacy for the patient. For the evaluation, we take the highest privacy level of patient C as the base and normalize everything with respect to this value. We use the following parameters for the simulation. The severities of patient A's and patient B's real SNPs are represented as truncated Gaussian random variables with $(\mu_A, \sigma_A) = (0.25, 0.15)$ and $(\mu_B, \sigma_B) = (0.75, 0.15)$, respectively. Furthermore, the severity of patient C's real SNPs are represented as a uniform distribution between 0 and 1. We also set $\mu(l) = 0.8$, $\sigma(l) = 0.25$, $\mu(k) = 2$, and $\sigma(k) = 0.75$. In Fig. 7, we illustrate the increase in privacy with increments in the storage redundancy for these three types of patients (A, B, and C). We observe that by increasing the storage redundancy, a patient with high severity real SNPs gains more privacy than a patient with lower severity real SNPs, hence the storage redundancy can be customized for each patient differently based on the types of his real SNPs. It can be argued that the amount of storage redundancy for a patient can leak information (to the curious party the SPU) about the severities of his real SNPs. However, the severity of the SNPs is not the only criteria to determine the storage redundancy for a desired level of genomic privacy as we discuss next.

Finally, we study the relationship between the severity of the real SNPs, the number of LD pairs per SNP (number of SNPs with which a particular SNP has LD, i.e., k), and the storage redundancy. We assign the V_i values of the real SNPs (in Υ_P) following a uniform distribution between 0 and 1. We set the LD parameters as $\mu(l) = 0.8$, $\sigma(l) = 0.25$, $\mu(k) = 2$, and $\sigma(k) = 1.5$. Then, we observe and compare the following potential scenarios in different types of patients: (i) The real low severity SNPs of the patient (i.e., his real SNPs with low V_i values) have a higher number of LD pairs (i.e., higher k values) with respect to his high severity real SNPs[9]; (ii) k values are assigned randomly to the SNPs; and (iii) the real high severity SNPs of the patient (i.e., his real SNPs with high V_i values) have a higher number of LD pairs (i.e., higher k values) with respect to his low severity real SNPs. Again, we set a patient's genomic privacy to 100% when the storage redundancy is maximum at the SPU (as in Method 1 in Section II-B). We illustrate our results in Fig. 8, and show different storage redundancy requirements for different types of patients (to provide the same level of privacy). For example, to achieve 40% genomic privacy, the SPU requires 400% storage redundancy for a patient whose less severe real SNPs have more LD pairs, whereas it requires 600% storage redundancy for another patient whose more severe real SNPs have more LD pairs (which means more storage per patient, as discussed in Section III-A). This result also supports our belief to customize the storage redundancy for each patient.

[9]We note that, in all cases, k values are obtained from the same truncated Gaussian distribution with $\mu(k) = 2$, and $\sigma(k) = 1.5$.

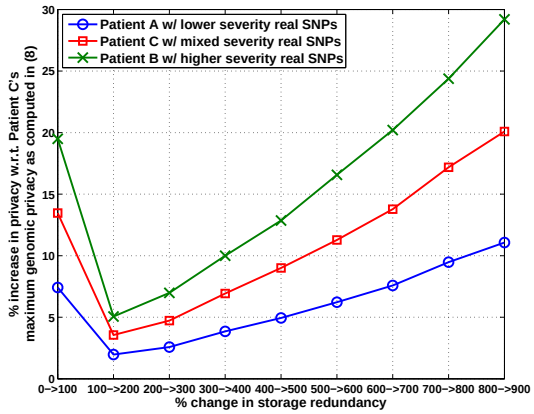


Fig. 7. Increase in genomic privacy of different types of patients with 100% increments in the storage redundancy. For example, increasing the storage redundancy from 400% to 500% would increase the privacy of Patient A (who carries mostly low severity real SNPs) by 5%, whereas the same scenario increases the privacy of Patient B (who carries mostly high severity SNPs) by 13%.

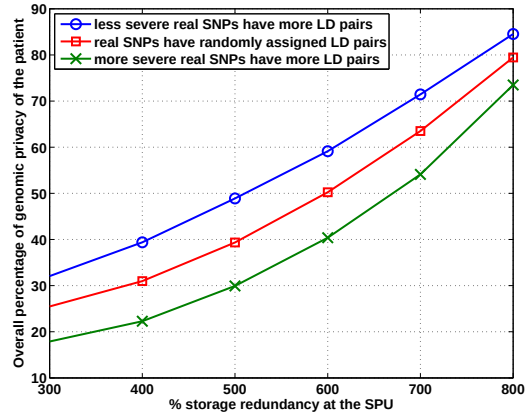


Fig. 8. Level of genomic privacy, as defined by (8), for different types of patients with varying storage redundancy.

We obtained similar patterns for further variations of the variables but we do not present these results due to the space limitation. In summary, depending on the actual $\mu(k)$, $\sigma(k)$, $\mu(l)$, $\sigma(l)$, and V_i values (which will be determined as a result of the medical research), the storage redundancy can be determined (and customized for each patient based on the types of his variations) for this approach to keep the genomic privacy of the patient at a desired level. Note that the curious party at the SPU cannot infer the real SNPs of the patient (or the severities of the patient's real SNPs) from the amount of customized storage redundancy, because the storage redundancy (for a desired level of genomic privacy) depends on various factors. For example, a patient with low storage redundancy (for a desired level of genomic privacy) could mean that (i) he carries mostly low severity real SNP (as in Fig. 7), (ii) he carries mixed severity real SNPs, but his less severe real SNPs have more LD pairs (as in Fig. 8), (iii) his real SNPs

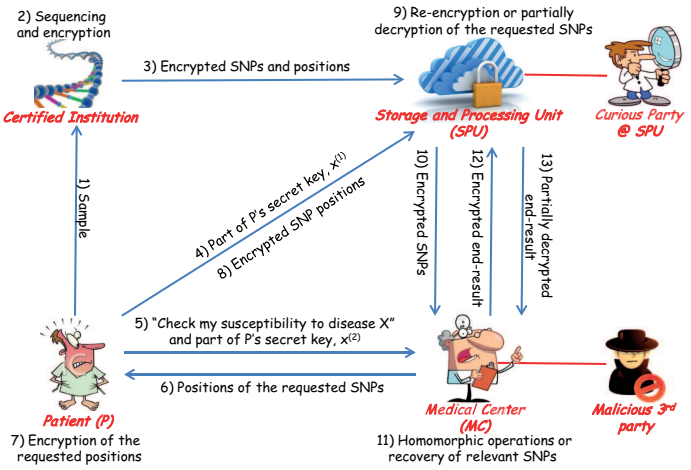


Fig. 9. Privacy-preserving protocol for disease-susceptibility test using Method 3.

(regardless of their severities) have low number of LD pairs (as in Fig. 3), or (iv) his real SNPs (regardless of their severities) have low LD strengths (as in Fig. 4).

D. Method 3: Encrypted Locations at the SPU

Let $\mathbb{L}^P = \{L_i : i \in \Upsilon_P\}$ represent the set of locations (on the DNA sequence) of the patient P's real SNPs (in Υ_P). As opposed to the previous two approaches, here, we propose to encrypt the locations of the SNPs along with their contents. By doing so, we save storage costs by storing only the real SNPs in Υ_P at the SPU (around 4 million) while providing the highest level of privacy (as in Section II-B). These benefits, however, come with a cost in the practicality of the algorithm, introducing extra steps for the patient (P) during the protocol. Although we can assume that these extra steps can easily be handled via the patient's smart card or mobile device, this approach still requires more message exchanges (as will be described next) between the parties, compared to the previous two approaches.

In some environments, randomly splitting the weak secret of the patient, and distributing two shares of the weak secret key to the SPU and MC might not be acceptable (e.g., when it is likely that the SPU and MC will collaborate to retrieve patient's weak secret). Therefore, for the sake of completeness, in the following, we present Method 3 with and without proxy encryption (i.e., without distributing the patient's secret to other parties).[10]

1) With Proxy Encryption: The initial steps of the protocol are the same as in Section II-B, except for Steps 2 and 3 in which the locations of the SNPs are encrypted and a Bloom filter [40] is generated. Below, we summarize the different steps of this approach (the unchanged steps are not repeated). These steps are illustrated in Fig. 9.

• **Step 2:** The Certified Institution (CI) first determines the locations of P's real SNPs (in Υ_P) and constructs $\mathbb{L}^P$. Then, the CI constructs a Bloom filter using the elements of $\mathbb{L}^P$ as inputs.

[10]Method 1 and Method 2 can also be modified similarly to avoid proxy encryption.

A Bloom filter is a simple space-efficient randomized data structure for representing a set in order to support membership queries [40]. A Bloom filter for representing a set $\mathbb{L}^P$ is described by an array of κ bits, initially all set to 0. It employs γ independent hash functions $\mathbb{H}_1, \ldots, \mathbb{H}_\gamma$ with range $\{1, \ldots, \kappa\}$. For every element $L_i \in \mathbb{L}^P$, the bits $\mathbb{H}_1(L_i), \ldots, \mathbb{H}_\gamma(L_i)$ in the array are set to 1. A location can be set to 1 multiple times, but only the first change has an effect.

After constructing the Bloom filter, the CI encrypts each element in $\mathbb{L}^P$ by using a symmetric key shared between the CI and P (established during Step 0 as in Section II-B) and generates $\mathbb{L}^P_E = \{\mathrm{E}(L_i) : i \in \Upsilon_P\}$. The CI also encrypts a "0" value (representing the potential SNPs in Ω_P) along with the real SNPs of the patient (using P's public key). Furthermore, the CI associates an arbitrary location L_0 for this "0" value and encrypts L_0 using the symmetric key between the CI and P to obtain $\mathrm{E}(L_0)$.

• **Step 3:** The CI sends the constructed Bloom filter and $\mathrm{E}(L_0)$ to the patient, and encrypted SNPs and locations to the SPU.

• **Step 6:** The MC tells the patient the locations of the SNPs that are required for the susceptibility test or requested directly as the relevant SNPs.

• **Step 7:** The patient inputs each requested location L_j to the Bloom filter to determine if the corresponding location is stored at his Bloom filter (i.e., to determine if he has a real SNP at the corresponding location).

To check if L_j belongs to $\mathbb{L}^P$, the patient checks whether all $\mathbb{H}_1(L_j), \ldots, \mathbb{H}_\gamma(L_j)$ are set to 1. If not, L_j definitely does not belong to $\mathbb{L}^P$. Otherwise, the patient assumes $L_j \in \mathbb{L}^P$, although this may be wrong with some probability. That is, a Bloom filter could yield a *false positive*, where it suggests that L_j is in $\mathbb{L}^P$ even though it is not. This probability can be decreased at the expense of increasing Bloom filter length (i.e., κ). Further, the false positive probability can be significantly reduced by using some proposed techniques such as [41], [42]. As a result of this process

(a) If the location is in his Bloom filter (i.e., if he has a real SNP at the corresponding location), P encrypts the location with the symmetric key between the CI and P.

(b) If the location is not in his Bloom filter (i.e., if he does not have a real SNP at the corresponding location), P uses $\mathrm{E}(L_0)$ as the encrypted location.

We note that the above operations can be easily done via the patient's smart card (e.g., by scanning the card at the MC as a consent to the test) or mobile device (e.g., by consenting via a smart phone application) by using the stored Bloom filter output, $\mathrm{E}(L_0)$, and symmetric key between the CI and P.

• **Step 8:** The patient sends the SPU the encrypted locations of the SNPs which will be provided to the MC.

• **Step 9:** The encrypted SNPs are sent to the MC in the same order as they are requested in Step 6.

(a) If only the end-result is requested, the corresponding SNPs are re-encrypted at the SPU under the patient's public key (re-encryption under the same public key is discusses in Section II-A). As there is only one value stored at the SPU representing the contents of the potential SNPs at which P does not have a variant (at location $\mathrm{E}(L_0)$), this value is re-encrypted for each different request of a non-variant, so that

the MC cannot infer the locations of the non-variants of the patient.

(b) If relevant SNPs are requested, the SPU partially decrypts the relevant SNPs by using a part of P's secret key following a proxy encryption protocol (Section II-A).

• **Step 10:** Re-encrypted (or partially decrypted) SNPs are sent to the MC by the SPU.

• **Step 11:** One of the following two scenarios occur at the MC:

(a) If only the end-result is requested, the MC computes P's total susceptibility to disease X by using the homomorphic properties of the Paillier cryptosystem (similar to the discussion in Section II-E)[11] under the patient's public key.

(b) If relevant SNPs are requested, the MC decrypts the message received from the SPU by using the other part of P's secret key and recovers the relevant SNPs.

• **Step 12:** The MC sends the encrypted end-result to the SPU.

• **Step 13:** The SPU partially decrypts the end-result using a part of P's secret key by following a proxy encryption protocol (Section II-A) and sends it back to the MC.

• **Step 14:** The MC decrypts the message received from the SPU by using the other part of P's secret key and recovers the end-result.

2) Without Proxy Encryption: In this approach, the SPU stores only the encrypted SNPs and encrypted locations. Genomic data encrypted by P's public key is only decrypted at P, and the weak secret of P remains only at P (i.e., shares of the weak secret are not distributed to the SPU or MC). Most of this approach is the same as Method 3 with proxy encryption. Indeed, the first 8 steps of the algorithm are the same, except for the distribution of parts of P's secret key. The only difference is the transfer of the end-result or the relevant SNPs to the MC as follows:

• If the relevant SNPs are requested by the MC, the SPU sends the encrypted SNPs (by P's public key) to P. P decrypts these SNPs (using his weak secret key) and sends them to the MC.

• If the end-result of the susceptibility test is requested by the MC, the disease-susceptibility test is done (via homomorphic operations) at the MC and the encrypted end-result is sent to P. Then, P decrypts the end-result and sends it back to the MC.

We note that the security of the communication between P and the MC is provided by symmetric keys as discussed before. The above operations put some more burdens on the patient during the protocol. However, we emphasize that these operations can be smoothly done on the patient's mobile device or smart card without requiring a substantial effort from the patient himself.

In summary, as the locations of the real SNPs are encrypted, a curious party at the SPU cannot infer the contents of the SNPs from their locations (as in Section II-B), hence it is enough to store only the real SNPs in Υ_P. Furthermore, the privacy provided by this approach (with or without proxy encryption) is the same as 900% redundancy in Method 2 (i.e., similar to Method 1), hence we do not discuss it again. Another advantage of this approach (i.e., Method 3 in general) is that individual contributions of the genetic variant markers remain secret at the MC, because the homomorphic operations are conducted at the MC. This advantage might become more significant when this approach is used for personalized medicine methods in which the pharmaceutical company (embodied in this case as the medical unit) does not want to reveal the genetic properties of its drugs. Thus, if introducing the described extra steps for the patient and few additional message exchanges between the parties are tolerated, this approach operates with relatively modest storage and yet provides very good privacy.

E. Computing Disease Susceptibility via Homomorphic Operations

We now present the disease-susceptibility test via homomorphic operations at the SPU for Method 1 (Section II-B) and Method 2 (Section II-C). Similar techniques can be used for Method 3 at the MC, as discussed in Section II-D.

The SPU uses a proper function to compute P's predicted disease susceptibility via homomorphic encryption. There are different functions for computing the predicted susceptibility. In [26], focusing on one example of many diseases that require a susceptibility test involving multiple SNPs, Kathiresan *et al.* propose to count the number of unfavorable alleles carried by the patient for each SNP related to a particular disease. Similarly, in [27], Ashley *et al.* propose to multiply the Likelihood Ratios (LRs) of the most important SNPs for a particular disease in order to compute a patient's predicted susceptibility.[12] Furthermore, a *weighted averaging* function can also be used, which computes the predicted susceptibility by weighting the contributions of SNPs by their contributions (e.g., LR values of the SNPs). Note that our proposed privacy-preserving mechanisms are not limited by the types of the functions (used to test the disease susceptibility). It is expected that these functions will evolve over time; hence the proposed algorithms can be developed to keep up with this evolution.

In the following, we discuss how to compute the predicted disease susceptibility at the SPU by using a toy example to show how the homomorphic encryption is used at the SPU. Initially, we assume that the function at the SPU is weighted averaging (which is an advanced version of the function proposed in [26]) and show how the predicted susceptibility is computed using encrypted SNPs. Then, we show how the function proposed in [27] (i.e., multiplication of LR values) can be utilized at the SPU.

1) Weighted Averaging: Assume that (for simplicity) the susceptibility to disease X is determined by the set of SNPs $\Omega = \{\mathrm{SNP}_m, \mathrm{SNP}_n\}$, which occur at particular locations of the DNA sequence.[13] The contributions of different states of SNP_i^P for $i \in \{m, n\}$ to the susceptibility to disease X are computed via previous studies (on case and control populations) and they are already known by the MC. That is, $\mathrm{p}_0^i(X) \triangleq \Pr(X|\mathrm{SNP}_i^P = 0)$ and $\mathrm{p}_1^i(X) \triangleq \Pr(X|\mathrm{SNP}_i^P = 1)$ ($i \in \{m, n\}$) are determined and known by the MC. Further, the contribution (e.g., LR value) of SNP_i to the susceptibility to disease X is denoted by C_i^X. Note that these contributions

[11] Although the discussion in Section II-E is held considering Method 1 (or Method 2), a similar technique is used for this approach at the MC, hence we do not discuss it again.

[12] LR values are determined as a result of medical studies.

[13] SNP_m^P and SNP_n^P are not necessarily among the real SNPs of the patient P (i.e., P does not need to have a variant at those locations).

are also computed by previous studies on case and control groups and they are known by the MC.

As we have discussed before, the SPU stores the set of SNPs of the patient P, encrypted by P's public key $(n, g, h = g^x)$.[14] Thus, the SPU uses $\mathrm{E}(\mathrm{SNP}_m^P, g^x)$ and $\mathrm{E}(\mathrm{SNP}_n^P, g^x)$ for the computation of predicted susceptibility of P to disease X. From now on, we drop the r values in the above encrypted messages for the clarity of the presentation (r values are chosen randomly from the set $[1, n/4]$ for every encrypted message as discussed in Section II-A). Similarly, the MC provides the following to the SPU in plaintext: (i) the markers for disease X (SNP_m and SNP_n), (ii) corresponding probabilities (p_j^i(X), $i \in \{m, n\}$ and $j \in \{0, 1\}$), and (iii) the contributions of each SNP (C_i^X).

Next, the SPU encrypts j ($j \in \{0, 1\}$) using P's public key to obtain $\mathrm{E}(0, g^x)$ and $\mathrm{E}(1, g^x)$ for the homomorphic computations.[15] Alternatively, we might assume that SNPs of a patient are stored at the SPU in pairs of $\{\mathrm{E}(|\mathrm{SNP}_i^P - 0|, g^x), \mathrm{E}(|\mathrm{SNP}_i^P - 1|, g^x)\}$ for each SNP_i^P, instead of the actual values of the SNPs. In this case, the above encryption at the SPU would not be required.

The SPU computes the predicted susceptibility of the patient P to disease X by using weighted averaging. This can be computed in plaintext as below:

$$\mathbb{S}_P^X = \frac{1}{C_m^X + C_n^X} \times \sum_{i \in m,n} C_i^X \left\{ \frac{\mathrm{p}_0^i(X)}{(0-1)} \left[\mathrm{SNP}_i^P - 1\right] + \frac{\mathrm{p}_1^i(X)}{(1-0)} \left[\mathrm{SNP}_i^P - 0\right] \right\}. \quad (9)$$

The computation in (9) can be realized using the encrypted SNPs of the patient (and utilizing the homomorphic properties of the Paillier cryptosystem) to compute the encrypted disease susceptibility, $\mathrm{E}(\mathbb{S}_P^X, g^x)$ as below:

$$\mathrm{E}(\mathbb{S}_P^X, g^x) = \left\{ \prod_{i \in \{m,n\}} \left\{ \left[\mathrm{E}(\mathrm{SNP}_i^P, g^x)\mathrm{E}(1, g^x)^{-1}\right]^{\Delta_i^1} \times \left[\mathrm{E}(\mathrm{SNP}_i^P, g^x)\mathrm{E}(0, g^x)^{-1}\right]^{\Delta_i^2} \right\}^{C_i^X} \right\}^{\Theta}, \quad (10)$$

where

$$\Delta_i^1 = \frac{\mathrm{p}_0^i(X)}{0-1}, \quad (11a) \qquad \Delta_i^2 = \frac{\mathrm{p}_1^i(X)}{1-0}, \quad (11b) \qquad \Theta = \frac{1}{C_m^X + C_n^X}. \quad (11c)$$

We note that the end-result in (10) is encrypted by P's public key.

Then, the SPU partially decrypts the end-result $\mathrm{E}(\mathbb{S}_P^X, g^x)$ using its share ($x^{(1)}$) of P's secret key (x) as discussed in Section II-A to obtain $\mathrm{E}(\mathbb{S}_P^X, g^{x^{(2)}})$ and sends it to the MC. Finally, the MC decrypts $\mathrm{E}(\mathbb{S}_P^X, g^{x^{(2)}})$ using its share ($x^{(2)}$) of P's secret key to recover the end-result $\mathbb{S}_P^X$.

[14]Encryption is done using the modified Paillier cryptosystem as discussed in Section II-A.

[15]This encryption can also be done at the MC and sent to the SPU.

In some genetic tests, the types of the real SNPs (e.g., homozygous or heterozygous) become also important. In this case, SNP_i^P can take three different values from the set $\{0, 1, 2\}$ to represent a potential SNP (i.e., non-variant), a real homozygous SNP, and a real heterozygous SNP, respectively. In such a scenario, to conduct the disease-susceptibility test via homomorphic operations, the SPU should store the squared values of the SNPs. That is, for each SNP_i^P of the patient P, the SPU should store $\mathrm{E}((\mathrm{SNP}_i^P)^2, g^x)$. Depending on the types of genomic tests that would be supported by the SPU (and the functions required for these tests), the format of storage of patient's SNPs can be determined beforehand, and SNPs can be stored accordingly just after the sequencing process.

2) Likelihood Ratio Test: We now assume that the predicted disease susceptibility is computed from the multiplication of Likelihood Ratios (LRs) of the corresponding SNPs as in [27] and show how such a computation would be handled at the SPU by using homomorphic operations.

In this approach, the predicted disease susceptibility is computed by multiplying the initial risk of the patient (e.g., for disease X) by the LR value of each SNP related to that disease (LR value of a SNP i depends on the value of SNP_i^P at the patient P). The initial risk of the patient P for the disease X is represented as I_X^P. We note that I_X^P is determined by considering several factors (other than patient's genomic data) such as patient's age, gender, height, weight, and environment. Thus, this initial risk can be computed directly by the MC. We also note that if the LR value corresponding to a particular SNP is less than one, the risk for the disease decreases. Otherwise, if the LR value is greater than one, the risk increases for the corresponding disease.

Similar to before, we assume that the susceptibility to disease X is determined by the set of SNPs in $\Omega = \{\mathrm{SNP}_m, \mathrm{SNP}_n\}$. We denote the LR values due to $\mathrm{SNP}_i^P = 0$ and $\mathrm{SNP}_i^P = 1$ for disease X as $L_X^i(0)$ and $L_X^i(1)$, respectively.

The SPU stores the SNPs of the patient P, encrypted by P's public key. The MC sends the following to the SPU: (i) $L_X^i(j)$ values ($i \in \{m, n\}$ and $j \in \{0, 1\}$) in plaintext, and (ii) the markers for disease X. The MC also encrypts the log of initial risk value, $\ln(I_X^P)$, by P's public key and sends $\mathrm{E}(\ln(I_X^P), g^x)$ to the SPU.[16]

The Paillier cryptosystem does not support multiplicative homomorphism in ciphertext (it only supports the multiplication of a ciphertext with a constant as discussed in Section II-A). Thus, instead of multiplying the LR values, we propose using addition in log-domain at the SPU. Thus, the SPU computes the predicted susceptibility of P to disease X as below:

$$\mathrm{E}(\ln(\mathbb{S}_P^X), g^x) = \mathrm{E}(\ln(I_X^P), g^x) \times \prod_{i \in m,n} \left\{ \left[\mathrm{E}(\mathrm{SNP}_i^P, g^x) \cdot \mathrm{E}(1, g^x)^{-1}\right]^{\Xi_i^1} \times \left[\mathrm{E}(\mathrm{SNP}_i^P, g^x) \cdot \mathrm{E}(0, g^x)^{-1}\right]^{\Xi_i^2} \right\}, \quad (12)$$

where

[16]Alternatively, the contribution of the initial risk to the disease susceptibility can be included to the end-result at the end, at the MC.

$$\Xi_i^1 = \frac{\ln(L_X^i(0))}{(0-1)}, \quad (13a) \qquad \Xi_i^2 = \frac{\ln(L_X^i(1))}{(1-0)}. \quad (13b)$$

We note that (12) corresponds to the below computation in plaintext:

$$\ln(\mathbb{S}_P^X) = \ln(I_X^P) + \sum_{i\in m,n} \Bigg\{ \left[\mathrm{SNP}_i^P - 1\right] \times \frac{\ln(L_X^i(0))}{(0-1)} + \left[\mathrm{SNP}_i^P - 0\right] \times \frac{\ln(L_X^i(1))}{(1-0)} \Bigg\}. \quad (14)$$

As before, the SPU partially decrypts $\mathrm{E}(\ln(\mathbb{S}_P^X), g^x)$ using $x^{(1)}$ (its share of P's secret key) to obtain $\mathrm{E}(\ln(\mathbb{S}_P^X), g^{x^{(2)}})$ and sends it to the MC. Finally, the MC decrypts $\mathrm{E}(\ln(\mathbb{S}_P^X), g^{x^{(2)}})$ using $x^{(2)}$ (its share of P's secret key) to recover $\ln(\mathbb{S}_P^X)$, and computes $e^{(\ln(\mathbb{S}_P^X))}$ to obtain $\mathbb{S}_P^X$. Similar to weighted averaging, if the types of the real SNPs are used for the test (in which there are three possible states for SNP_i^P), squared values of the SNPs should be stored at the SPU for each patient.

III. Evaluation and Implementation of the Proposed Methods

In Fig. 10, based on the discussion in the previous sections, we graphically compare the proposed methods considering the level of privacy they provide, their practicality (for the patient), and their storage requirements (at the SPU). In this section, we report our findings about the complexity and security of the proposed methods.

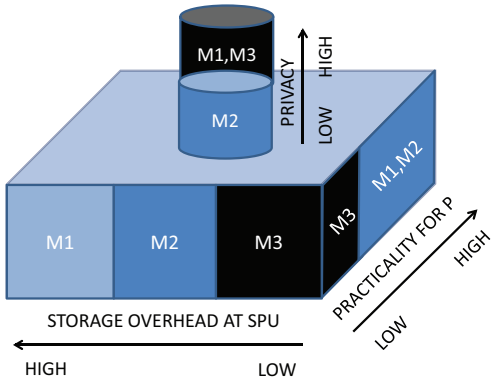


Fig. 10. Privacy, practicality, and storage overhead comparison of the proposed methods.

A. Implementation and Complexity Evaluation

To evaluate the practicality of the proposed privacy-preserving algorithms, we implemented them, and assessed their storage requirements and computational complexities on Intel Core i7-2620M CPU with 2.70 GHz processor under Windows 7 Enterprise 64-bit Operating System. We set the size of the security parameter (n in Paillier cryptosystem in Section II-A) to 1024 bits. We computed the disease susceptibility using weighted averaging (at the SPU or MC, see Section II-E1)[17] and real SNP profiles from [43]. Our implementation relies on a MySQL 5.5 database managed by the open source tool MySQL Workbench. To provide a platform-independent implementation, we used the Java programming language along with the open-source Integrated Development Environment, NetBeans IDE 7.1.1., for the implementation of the Java code.[18] In Fig. 11, we illustrate three screen shots from our implementation of Method 1 in which we illustrate the operations conducted at the Patient (P), Storage and Processing Unit (SPU), and Medical Center (MC), respectively.[19]

In Table II, we summarize the computational and storage complexities of the proposed methods at (i) Certified Institution (CI), (ii) SPU, (iii) MC, and (iv) P. We evaluate the proposed methods considering the following costs: (i) encryption of patient's variants, (ii) disease-susceptibility test at the SPU via homomorphic operations (using ten variants), (iii) decryption of the end-result (or relevant SNPs), (iv) proxy encryption, and (v) storage costs, in which ϑ represent the percentage of storage redundancy at the SPU. We did not explicitly implement the Bloom filter (for Method 3) and symmetric enryption/decryption between the parties for the security of the communication. However, the computational costs due to these operations are negligible compared to Paillier encryption/decryption and homomorphic operations.

We emphasize that the encryption of the variants at the CI is a one-time operation and is significantly faster than the sequencing and analysis of the sequence (which takes days). Further, this encryption can be conducted much more efficiently by computing some parameters, such as (g^r, h^r) pairs, offline for various r values, for each patient. Indeed, by computing (g^r, h^r) pairs offline, we observe that the encryption takes only 0.017 ms per variant at the CI.

It is also possible to conduct private statistical tests (by a medical researcher) on the data stored at the SPU in order to get statistics about the variants of multiple patients. Conducting such a statistical test for a variant (about its type) on 100K patients takes around 55 minutes at the SPU and scales linearly with the number of patients. Note that such a statistical test is only possible with Method 1 or Method 2; using Method 3 and querying the encrypted locations of SNPs from 100K patients is not practical for this application.

In summary, all these numbers show the practicality of our privacy-preserving algorithms.

B. Security Evaluation

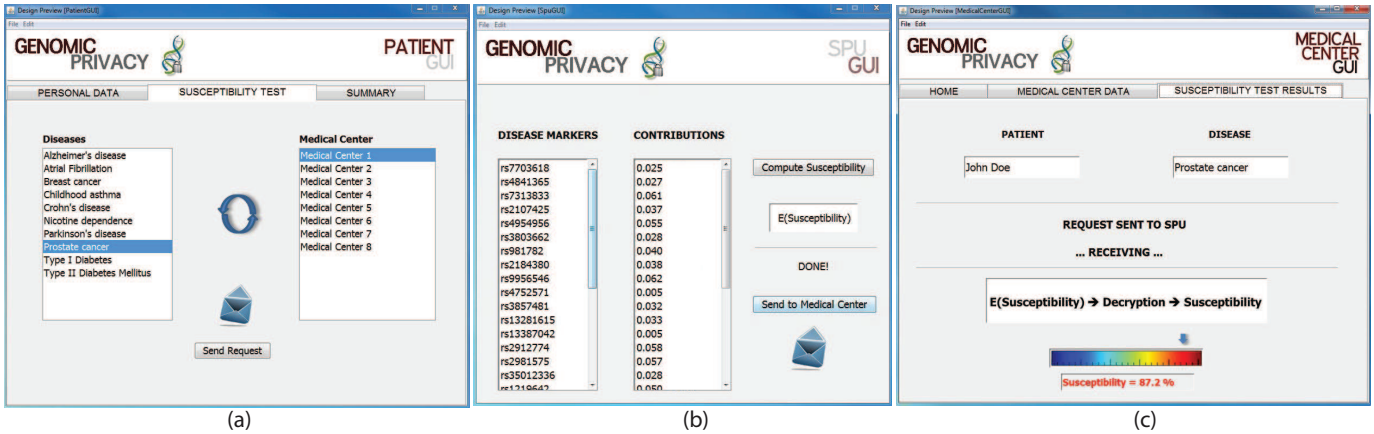
The proposed schemes preserve the privacy of patients' genomic data relying on the security strength of modified Paillier cryptosystem (in Section II-A). The extensive security evaluation of the modified Paillier cryptosystem can be found in [34]. Below we summarize two important security features of this cryptosystem.

• One-wayness: This property means that no efficient adversary has any significant chance of finding a pre-image to the ciphertext when he sees only the ciphertext and the public key of the patient. It is shown in [34] that the one-wayness

[17] LR test in Section II-E2 also has similar complexity.

[18] We note that our code for the implementation is not optimized, and better results can be expected with an optimized implementation.

[19] We skip the illustration of the intermediate steps (discussed in Section II-B), and only present the key steps of the algorithm.



(a) (b) (c)

Fig. 11. Implementation of Method 1 at P, the SPU, and MC. In Fig. 11(a), P selects the type of susceptibility test and the MC, which will conduct the test. Next, in Fig. 11(b), the SPU conducts the disease-susceptibility test using the encrypted SNPs of P and the markers it received from the MC, and sends the encrypted end-result to the MC. Finally, in Fig. 11(c), the MC receives the encrypted end-result from the SPU, decrypts it using its share of P's secret key, and obtains the plaintext end-result of the test. That is, the MC recovers the probability that P will develop "prostate cancer" in the future based on his genetic variations.

Method 1 and Method 2				
@CI	**@SPU**			**@MC**
Paillier Encryption	Homomorphic Operations	Proxy Encryption	Storage	Paillier Decryption
30 ms./variant	1 sec. (10 variants)	2 ms.	$500 \times \left(1 + \frac{\vartheta}{100}\right)$ MB/patient	26 ms.
Method 3 with proxy encryption				
@CI	**@SPU**		**@MC**	
Paillier Encryption	Proxy Encryption	Storage	Homomorphic Operations	Paillier Decryption
30 ms./variant	2 ms.	500 MB/patient	1 sec. (10 variants)	26 ms.
Method 3 without proxy encryption				
@CI	**@SPU**	**@MC**	**@P**	
Paillier Encryption	Storage	Homomorphic Operations	Paillier Decryption	
30 ms./variant	500 MB/patient	1 sec. (10 variants)	26 ms.	

TABLE II
COMPUTATIONAL AND STORAGE COMPLEXITIES OF THE PROPOSED METHODS

of the modified Paillier cryptosystem can be related to the Lift Diffie-Hellman problem which is shown to be as hard as the partial Discrete Logarithm problem.

• Semantic security: This property ensures that an adversary will be unable to distinguish pairs of ciphertexts based on the message they encrypt. It is shown in [34] that if Decisional Diffie-Hellman Assumption (a computational hardness assumption about a certain problem involving discrete logarithms in cyclic groups) in $\mathbb{Z}^*_{n^2}$ holds, then the modified Paillier cryptosystem is semantically secure.

Finally, if the weak secret of the patient, x, is randomly divided and distributed to the Storage and Processing Unit (SPU) and Medical Center (MC) as in Method 1, this weak secret could be revealed if the MC colludes with the SPU, but the factors n, p, and q remain secret. We note that such a collusion is not considered in this study. However, for the sake of completeness, in Section II-D2, we present an alternative approach (Method 3 without proxy encryption) that avoids distributing the patient's weak secret to other parties, hence is robust against such a collusion.

IV. CONCLUSION AND FUTURE WORK

In this paper, we have introduced privacy-preserving schemes for the utilization of genomic data in medical tests and personalized medicine methods. We have proposed new models based on the existence of a Storage and Processing Unit (SPU) between the patient and the medical unit (e.g., healthcare center or pharmaceutical company). We have shown that encrypted genomic data of the patients can be stored at the SPU and processed (for medical tests and personalized medicine methods) using homomorphic encryption and proxy encryption. For this purpose, we utilize a variation of the Paillier cryptosystem to encrypt and process the genomic data. We have proposed different techniques for the storage and process of the SNPs at the SPU while preserving the patient's privacy. Moreover, we analyzed the relationship between the storage cost, privacy of the patient, strength of relationship between the genetic markers, and the characteristics of the markers. This analysis could play a key role for customizing the storage redundancy of the genomic data for each patient, while keeping the privacy of the patient at a desired level. We also implemented the proposed schemes and showed their

efficiency and practicality through a complexity evaluation. We are confident that our proposed privacy-preserving schemes will encourage the use of genomic data, by the individual and by the medical unit, and accelerate the move of genomics into clinical practice.

The extension of this work opens the doors to various exciting research opportunities that we briefly discuss in the following. One of the genomic privacy risks is that several SNPs reveal more than one disease. Even if the SPU reveals only the end-results, the SNPs used to test a benign disease might overlap with the SNPs used to test a serious disease. Thus, a medical center could obtain information about the susceptibility of a patient to a serious disease by checking his susceptibility to a benign disease. Therefore, this issue should also be considered along with the Linkage Disequilibrium (LD) between the SNPs before using them in medical tests.

Furthermore, as the number of SNPs used increases, the accuracy of the medical test also increases –up to a point– and the error in accuracy begins to increase when excessive numbers of SNPs (irrelevant information) are used. Whereas, releasing too many SNPs is a risk for a person's genomic privacy. Consequently, there is a tradeoff between (i) the accuracy of the medical test, (ii) the number of SNPs used, and (iii) privacy. Eventually, it will be important to determine the optimal number of SNPs and the types of SNPs to be used for various medical tests in order to (i) maximize accuracy, (ii) maximize privacy (minimize data leak), (iii) minimize complexity, and (iv) find an optimal point between these three factors.

REFERENCES

[1] A. Cavoukian, "Privacy by design," 2009, http://www.ontla.on.ca/library/repository/mon/23002/289982.pdf.

[2] S. F. Gurses, "Multilateral privacy requirements analysis in online social network services," 2010, PhD thesis, KU Leuven.

[3] M. Langheinrich, "Principles of privacy-aware ubiquitous systems," *Proceedings of Ubiquitous Computing (UbiComp)*, 2001.

[4] G. van Blarkom, J. Borking, and J. Olk, "Handbook of privacy and privacy-enhancing technologies (the case of intelligent software agents)," *College bescherming persoonsgegevens*, 2003.

[5] http://www.personalgenomes.org/consent/PGP_Consent_Approved_02212012.pdf, Visited on 29/Oct/2012.

[6] http://www.genomethics.org, Visited on 29/Oct/2012.

[7] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik, "Privacy preserving error resilient DNA searching through oblivious automata," *CCS '07: Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 519–528, 2007.

[8] M. Blanton and M. Aliasgari, "Secure outsourcing of DNA searching via finite automata," *DBSec'10: Proceedings of the 24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy*, pp. 49–64, 2010.

[9] S. Jha, L. Kruger, and V. Shmatikov, "Towards practical privacy for genomic computation," *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pp. 216–230, 2008.

[10] F. Bruekers, S. Katzenbeisser, K. Kursawe, and P. Tuyls, "Privacy-preserving matching of DNA profiles," Tech. Rep., 2008.

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

[12] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik, "Countering GATTACA: Efficient and secure testing of fully-sequenced human genomes," *CCS '11: Proceedings of the 18th ACM Conference on Computer and Communications Security*, pp. 691–702, 2011.

[13] M. Canim, M. Kantarcioglu, and B. Malin, "Secure management of biomedical data with cryptographic hardware," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 1, 2012.

[14] D. Eppstein, M. T. Goodrich, and P. Baldi, "Privacy-enhanced methods for comparing compressed DNA sequences," *CoRR*, vol. abs/1107.3593, 2011. [Online]. Available: http://arxiv.org/abs/1107.3593

[15] D. Eppstein and M. T. Goodrich, "Straggler identification in round-trip data streams via Newton's identities and invertible Bloom filters," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 2, pp. 297–306, 2011.

[16] R. Wang, Y. F. Li, X. Wang, H. Tang, and X. Zhou, "Learning your identity and disease from research papers: Information leaks in genome wide association study," *CCS '09: Proceedings of the 16th ACM Conference on Computer and Communications Security*, pp. 534–544, 2009.

[17] B. Malin and L. Sweeney, "How (not) to protect genomic data privacy in a distributed network: Using trail re-identification to evaluate and design anonymity protection systems," *Journal of Biomedical Informatics*, vol. 37, pp. 179–192, Jun. 2004.

[18] N. Homer, S. Szelinger, M. Redman, D. Duggan, and W. Tembe, "Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays," *PLoS Genetics*, vol. 4, Aug. 2008.

[19] J. Gitschier, "Inferential genotyping of Y chromosomes in Latter-Day Saints founders and comparison to Utah samples in the HapMap project," *Am. J. Hum. Genet.*, vol. 84, pp. 251–258, 2009.

[20] X. Zhou, B. Peng, Y. F. Li, Y. Chen, H. Tang, and X. Wang, "To release or not to release: Evaluating information leaks in aggregate human-genome data," *ESORICS'11: Proceedings of the 16th European Conference on Research in Computer Security*, pp. 607–627, 2011.

[21] S. E. Fienberg, A. Slavkovic, and C. Uhler, "Privacy preserving GWAS data sharing," *Proceedings of the IEEE 11th International Conference on Data Mining Workshops (ICDMW)*, Dec. 2011.

[22] Y. Chen, B. Peng, X. Wang, and H. Tang, "Large-scale privacy-preserving mapping of human genomic sequences on hybrid clouds," *NDSS'12: Proceeding of the 19th Network and Distributed System Security Symposium*, 2012.

[23] R. Wang, X. Wang, Z. Li, H. Tang, M. K. Reiter, and Z. Dong, "Privacy-preserving genomic computation through program specialization," *Proceedings of the 16th ACM Conference on Computer and Communications Security*, pp. 338–347, 2009.

[24] R. Agrawal, A. Evfimievski, and R. Srikant, "Information sharing across private databases," *Proceedings of SIGMOD Conference*, 2003.

[25] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung, "Efficient robust private set intersection," *Proceedings of the 7th International Conference on Applied Cryptography and Network Security*, pp. 125–142, 2009.

[26] S. Kathiresan, O. Melander, D. Anevski, C. Guiducci, and N. Burtt, "Polymorphisms associated with cholesterol and risk of cardiovascular events," *The New England Journal of Medicine*, vol. 358, pp. 1240–1249, 2008.

[27] E. Ashley, A. Butte, M. Wheeler, R.Chen, and T. Klein, "Clinical assessment incorporating a personal genome," *The Lancet*, vol. 375, no. 9725, pp. 1525–1535, 2010.

[28] S. Seshadri, A. Fitzpatrick, M.A. Ikram, A. DeStefano, V. Gudnason, M. Boada, J. Bis, A. Smith, M. Carassquillo, J. Lambert, C. Consortium, G. Consortium, and E. Consortium, "Genome-wide analysis of genetic loci associated with Alzheimer disease," *JAMA*, vol. 303, pp. 1832–1840, 2010.

[29] http://www.ncbi.nlm.nih.gov/projects/SNP/, Visited on 29/Oct/2012.

[30] D. Greenbaum, A. Sboner, X. Mu, and M. Gerstein, "Genomics and privacy: Implications of the new reality of closed data for the field," *PLoS Computational Biology*, vol. 7, no. 12, 2011.

[31] M. Raykova, H. Zhao, and S. M. Bellovin, "Privacy enhanced access control for outsourced data sharing," *Financial Cryptography and Data Security*, 2012.

[32] M. T. Goodrich and M. Mitzenmacher, "Privacy-preserving access of outsourced data via oblivious RAM simulation," *Proceedings of the 38th International Conference on Automata, Languages and Programming - Volume Part II*, pp. 576–587, 2011.

[33] E. Stefanov, E. Shi, and D. Song, "Towards practical oblivious RAM," *NDSS'12: Proceeding of the 19th Network and Distributed System Security Symposium*, 2012.

[34] E. Bresson, D. Catalano, and D. Pointcheval, "A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications," *Proceedings of Asiacrypt 03, LNCS 2894*, pp. 37–54, 2003.

[35] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, "Secure attribute-based systems," *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pp. 99–112, 2006.

[36] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security*, vol. 9, pp. 1–30, Feb. 2006.

[37] D. S. Falconer and T. F. Mackay, *Introduction to Quantitative Genetics (4th Edition)*. Harlow, Essex, UK: Addison Wesley Longman, 1996.
[38] C. Diaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," *Proceedings of Privacy Enhancing Technologies Symposium (PETS)*, 2002.
[39] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," *Prioceedings of Privacy Enhancing Technologies Symposium (PETS)*, 2002.
[40] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *ACM Communications*, vol. 13, no. 7, pp. 422–426, 1970.
[41] F. Hao, M. Kodialam, and T. V. Lakshman, "Building high accuracy Bloom filters using partitioned hashing," *Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems*, pp. 277–288, 2007.
[42] P. S. Almeida, C. Baquero, N. Preguiça, and D. Hutchison, "Scalable Bloom filters," *Information Processing Letters*, vol. 101, no. 6, pp. 255–261, 2007.
[43] The 1000 Genomes Project Consortium, "A map of human genome variation from population-scale sequencing," *Nature*, vol. 467, pp. 1061–1073, 2010.