Dynamic Privacy-Preserving Genomic Susceptibility Testing

Mina Namazi
Signal Theory and Communication Department
University of Vigo
36310 Vigo, Spain
mnamazi@gts.uvigo.es

Juan Ramón Troncoso-Pastoriza
Signal Theory and Communication Department
University of Vigo
36310 Vigo, Spain
troncoso@gts.uvigo.es

Fernando Pérez-González
Signal Theory and Communication Department
University of Vigo
36310 Vigo, Spain
fperez@gts.uvigo.es

ABSTRACT

The field of genomic research has considerably grown in the recent years due to the unprecedented advances brought about by Next Generation Sequencing (NGS) and the need and increasing widespread use of outsourced processing. But this rapid increase also poses severe privacy risks due to the inherently sensitive nature of genomic information. In this work, we address privacy-preserving genetic susceptibility tests outsourced to an untrustworthy party, enhancing previous approaches in terms of computation and communication efficiency by leveraging the use of somewhat homomorphic lattice encryption and relinearization operations to achieve more efficient constructions. Additionally, we also propose a more general construction which deals with several different medical units (such as pharmaceutical companies or hospitals), managing patients' consent to the disclosure of test results for each of these units, which may dynamically join the system. Our scheme features an attribute-based homomorphic cryptosystem which enables enforcing the patient's access policy referred to the different medical units.

Keywords

Privacy Preserving Techniques; Genomic Privacy; Lattice-Based Cryptography; Attribute Based Encryption

1. INTRODUCTION

The advent of Next Generation Sequencing enables an increasing use of genomic data in various domains such as health care, biomedical research, disease susceptibility tests, and forensics criminal investigations [10]. Nowadays, genomic data can be inexpensively sequenced and stored in a digital format, becoming widely available for the aforementioned applications, and rapidly surpassing the computation capacity of in-house infrastructures in medical centers and laboratories. Hence, outsourcing genomic processing arises as a necessity to cope with the volume of sequenced data.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

IH&MMSec 2016, June 20–23, 2016, Vigo, Spain.
© 2016 ACM. ISBN 978-1-4503-4290-2/16/06…$15.00
DOI: http://dx.doi.org/10.1145/2909827.2930791

The benefits of a widespread use and study of genomic data in advancing medicine research are unquestionable, but the inherently sensitive and identifiable nature of the genome entails severe privacy risks. Whenever a patient reveals her sequenced genomic data to a genomic center for running some tests, she loses control over these data and over the amount of leaked personal information: the lab has access to more information than just the test results, therefore harming the patient's privacy. Moreover, the genomic information can be linked to ancestors and relatives of an individual, so its leakage also affects their privacy. These issues are aggravated when the sequences are outsourced to an untrustworthy environment, like a Cloud service, for their processing. Outsourcing genomic data exposes them to the service and infrastructure providers and makes them vulnerable to attacks and accesses violating patients' privacy.

Consequently, the rapidly decreasing cost of DNA sequencing which pushes forward the availability of genomic test services also evidences the need of technological means for protecting these sequences from unauthorized access and processing. These privacy issues have motivated recent proposals of mechanisms for performing outsourced calculations on genomic sequences in a privacy-preserving way. In this work we focus on one of the most recent privacy-preserving mechanisms for disease susceptibility outsourced processing by Ayday *et al.* [2], and enhance it with two main contributions: a) Our proposed scheme features lattice-based somewhat homomorphic encryption in order to enable fully outsourced processing in the service provider, with no intermediate interaction needed by the medical center or the patient for running the susceptibility tests, and moves the bulk of data processing to the outsourced environment, which has enough computational power to operate on big data without having access to the original DNA of the patient; b) we generalize our construction to allow for different medical centers with variable access structures to the data, controlled and defined by the patient, in static and dynamic configurations.

1.1 Notation and Structure

We use calligraphic $\mathcal{L}$ letters when referring to the participants in the protocols. Uppercase letters denote matrices and lowercase letters denote elements from a polynomial ring; boldface letters denote vectors of polynomials; $a \cdot b$ denotes a polynomial product, and $\boldsymbol{a} \cdot \boldsymbol{b}$ an scalar product between vectors of polynomials. The subindex $\boldsymbol{x}_{E_P}$ denotes the result of the encryption of x with the key belonging to $\mathcal{P}$. The key owner will be omitted when there is no ambiguity.

The rest of the paper is organized as follows: Section 2 introduces some recent related work. Section 3 revises some concepts used for our proposed constructions. In Section 4 we revise Ayday *et al.*'s scheme and present our enhancements based on somewhat homomorphic lattice encryption. Section 5 sketches our proposed extended scheme, with static and dynamic management of several medical units. Finally, Section 6 draws some conclusions and hints for future work.

2. RELATED WORK

The recent privacy-preserving genomic works can be classified into three main categories: The first category deals with *private string searching and comparing*, initiated by Troncoso-Pastoriza *et al.* [13]; within this category, Chen *et al.* [6] recently proposed a private read mapping protocol, aligning short DNA sequences to human DNA with a secure and efficient algorithm.

The second category tackles *private release of aggregated data*; Wang *et al.* [14] have recently proposed a secure and privacy-preserving method to find homologous genes.

The third category, to which this work belongs, deals with *private clinical gemomics*, the field that most directly impacts citizens and their need for immediate privacy. Within this category, Baldi *et al.* [3] proposed a medical tool for supporting privacy preserving string comparison methods for individual medical tests over genomic data, and Canim *et al.* [5] applied hardware encryption for efficiently processing biomedical data; finally, one of the most representative and recent works has been proposed by Ayday *et al.* [2], which addresses secure medical tests of certain disease susceptibility on patients' genomic data.

For this purpose, they claim that existing secure searching methods do not provide a proper framework for various types of genomic tests to develop proper personalized medical methods, and they propose to use a trusted party, who helps in sequencing, generating and distributing secrets required for the system among the other parties. Additionally, their solution is outsourced, featuring a storing and processing unit ($\mathcal{SPU}$) between the patient and the medical center, in such a way that the calculations are shared between the medical center and the $\mathcal{SPU}$ through an additive homomorphic encryption and a partial decryption process, relieving the patient from most of the computations.

We advance and enhance this proposal with two main contributions: we enable that the $\mathcal{SPU}$ can execute all the heavy computations needed for operating on encrypted data, without the need to interact with the medical center, which only has to decrypt the results; for this purpose, we introduce a lattice-based somewhat homomorphic encryption which enables our fully secure outsourced protocol.

Furthermore, Ayday's scheme allows neither that different medical units can join the protocol, nor that the patient updates her consent on the tests that can be performed on her data, therefore obliging her to re-upload the whole sequenced data if any change occurs. Our second contribution enables more than one medical unit through different patient-defined access rights coded as attributes; to this aim, we apply attribute-based encryption (ABE) over lattices with homomorphic properties, and we also sketch a dynamic scheme which supports new medical centers to join the system and a dynamic modification of the access policies.

3. PRELIMINARIES AND CRYPTOSYSTEM

For the sake of completeness, we present some concepts and background needed for later describing our proposed schemes. We briefly revise the background on genomic sequences needed to implement a susceptibility test, and also the structure of Bloom filters; finally, we present a somewhat homomorphic cryptosystem based on the Ring Learning with Errors hardness problem, and the primitives we use for building our privacy-preserving susceptibility protocol.

3.1 Genomic Background

Every human being's aligned genomic sequence has over 3 billion base pairs of short reads [1] sampled randomly, comprising between 100 and 400 nucletoides each.

After being sequenced, the position of the first aligned nucleotide in a short read content is denoted as the short read's position with respect to the reference genome, in the form $L_{i,j} = \langle x_i | y_j \rangle$, where $x_i \in [1, 23]$ represents the chromosome and $y_j \in [1, 2.4 \cdot 10^8]$ represents the position of the short read within chromosome x_i. Due to mutations, sequencing errors and inheritance, a patient's sequence differs from the reference genome. A *Cigar String* represents these variants as pairs of nucleotide lengths and the associated operations (see Fig. 1). The most common and relevant variants are called SNPs (Single Nucleotide Polymorphisms); they represent positions in the genome holding a nucleotide that varies between individuals. SNPs are particularly suitable for running susceptibility tests of certain diseases. Weighted averaging [8] or Likelihood Ratio (LR) tests [12] are commonly used to measure the susceptibility to a given disease; we give the formula for these tests in Section 4.

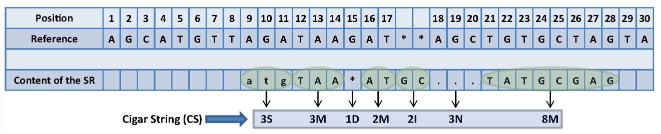

Figure 1: Short read aligned to the reference [1].

There are over 100 million different SNPs recognized in the human population (and growing), with every individual carrying around 4 million of them. The positions of these 4 million SNPs are different in each individual. Following a similar notation as in [2], we refer to these set of SNPs as "real SNPs" and the remaining ones as "potential SNPs"; the i-th SNP for patient $\mathcal{P}$ is represented as $\text{SNP}^{P,i}$, where $\text{SNP}^{P,i} = 1$ if $\mathcal{P}$ has a real SNP at this location, and $\text{SNP}^{P,i} = 0$ otherwise[1]; Γ^P denotes the set of positions for real SNPs of patient $\mathcal{P}$ (at which $\text{SNP}^{P,i} = 1$), and γ^P the set of positions of potential SNPs, at which $\text{SNP}^{P,i} = 0$.

3.2 Bloom Filters

The problem of calculating a susceptibility function mainly deals with appropriately weighting the SNPs which have some contribution in the corresponding disease, depending

[1]Besides the presence or absence of a SNP, the formulation can be extended to consider the type of SNP (homozygous or heterozygous), encoded as $\{0, 1, 2\}$ instead of just $\{0, 1\}$. We refer the interested reader to [2] for further details.

on whether they are present (or absent) in the patient. Determining whether they are present or not can be reduced to a membership problem in the sets Γ^P and γ^P.

Ayday *et al.* [2] propose to use Bloom Filters as efficient data structures to support set membership queries in a set L^P of ordered elements represented as an array of n bits, where a small probability of false positives is allowed. The Bloom filter is built as follows: all the n bits are initially set to 0; the generator can then define κ independent hash functions ($\kappa \ll n$), $H_1, H_2, ..., H_\kappa$ with range $\{0, n-1\}$. For each element of the set Γ^P, the κ hash functions will output κ (possibly repeated) bit positions in $\{0, n-1\}$, which are set to one in the Bloom Filter bit string; the filter is eventually composed of the final n-bit string and the κ hash functions. In order to check for membership of an element i in L^P, anyone with the filter can run the hash functions on i; if any of them hashes to a position where the filter bit string has a 0, i does not belong to Γ^P.

3.3 RLWE-based Homomorphic Encryption

RLWE (*Ring Learning with Errors*) is the hardness problem in which the most recent lattice-based cryptosystems are based; it is an efficient algebraic variation of Learning With Errors (LWE) introduced by Lyubaskevsky *et al.* [9] as an indistinguishability problem between two pairs of values: for a security parameter λ, let $f(x) = x^d + 1$ where $d = d(\lambda)$ is a power of 2. Let $q = q(\lambda) \geq 2$ be an integer, $R = \mathbb{Z}[x]/(f(x))$ and $R_q = R/qR$ two polynomial quotient rings. Let $\chi = \chi(\lambda)$ be an error distribution over R. For the secret $s \in R_q$, the RLWE distribution produces (a_i, b_i) such that a_i is sampled uniformly from R_q, and $b_i = a_i \cdot s + e_i$ with $e_i \leftarrow \chi$. RLWE assumes that this distribution in indistinguishable from sampling (a_i, b_i) uniformly in R_q^2.

Some of the most recent and efficient homomorphic cryptosystems are based on RLWE. We will focus here only on leveled Somewhat Homomorphic Encryption (SHE), which holds the necessary properties to achieve our purposes. An SHE cryptosystem comprises four functions $\texttt{SHSetup}$, $\texttt{SHGen}$, $\texttt{SHEnc}$, $\texttt{SHDec}$; we exemplify them with the scheme by Brakersky *et al.* [4] adapted to non-binary plaintexts:

$\texttt{SHSetup}(1^\lambda) \rightarrow pp_{SW}$: This function outputs the public parameters $pp_{SW} = (d, q, \chi)$, where q is a module, $d = d(\lambda)$ is the lattice dimension, and $\chi = \chi(\lambda)$ is an error distribution such that RLWE is secure against attacks with at least 2^λ as security parameter.

$\texttt{SHGen}(pp_{SW}) \rightarrow (\boldsymbol{sk}_u, \boldsymbol{pk}_u)$: It generates a public-secret key pair for user u. For this purpose, it samples randomly $s' \leftarrow \chi$, and outputs $\boldsymbol{sk}_u = \boldsymbol{s} = (1, s') \in R_q^2$. In order to generate the public key, two polynomials $a \leftarrow R_q$ and $e \leftarrow \chi$ are sampled randomly, such that $\boldsymbol{pk}_u = (b = a \cdot s' + te, -a)$ with $t \in \mathbb{Z}$ being the allowed plaintext cardinality. It must be noted that the public key is just an encryption of zero.

$\texttt{SHEnc}(pp_{SW}, \boldsymbol{pk}_u, m) \rightarrow \boldsymbol{m}_E$: To encrypt a message $m \in R_t$, randomly sample $r \leftarrow \chi$ and $\boldsymbol{e} \leftarrow \chi^2$. The encryption of m is $\boldsymbol{m}_E = (m, 0) + t \cdot \boldsymbol{e} + \boldsymbol{pk}_u \cdot r \in R_q^2$.

$\texttt{SHDec}(pp_{SW}, \boldsymbol{s}, \boldsymbol{m}_E) \rightarrow m$: It outputs the decypted $m = [[\boldsymbol{m}_E \cdot \boldsymbol{s}]_q]_t$, where $[.]_t$ is the modular reduction modulo t.

For the sake of clarity, we will omit the pp_{SW} argument in the encryption/decryption functions from now on. This somewhat homomorphic cryptosystem can homomorphically evaluate bounded polynomials of (sublinear) degree N (only logarithmic depth), at the cost of an increase of the ciphertext noise after each homomorphic operation, therefore increasing the decryption error rate of the ciphertex as the polynomial factor grows per operation. Addition of plaintexts is equivalent to adding two ciphertext pairs ($\boldsymbol{m}'_E = \boldsymbol{m}_E^{(1)} + \boldsymbol{m}_E^{(2)}$)), and multiplication of two plaintexts is homomorphic to the tensor product between the two ciphertexts ($\boldsymbol{m'_E} = \boldsymbol{m}_E^{(1)} \otimes \boldsymbol{m}_E^{(2)}$), which becomes a three-dimensional ciphertext that can be decrypted with the secret key $\boldsymbol{s}'' = (1, s', s' \cdot s')$ (for further details, we refer the reader to [4]).

A homomorphic multiplication increases the dimension of the ciphertext, so Brakerski *et al.* introduced modulus switching and key switching techniques working in a leveled chain of keys $\{\boldsymbol{s}_0, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_N\}$ to tackle the problem. Without going into details which are out of the scope of this paper, the combined techniques allow for a reduction of an expanded three-dimensional ciphertext back into a regular two-dimensional ciphertext, with the help of a relinearization matrix $\boldsymbol{B}$ which holds encryptions of pieces of $\boldsymbol{s}''$ under the destination key $\boldsymbol{s}_2$.

While this process is originally intended as a dimension reduction to allow for the homomorphic evaluation of higher degree polynomials, the relinearization step can be used in a more general way, as a proxy re-encryption, in order to change a ciphertext between a key $\boldsymbol{s}_{u_1}$ to another ciphertext under key $\boldsymbol{s}_{u_2}$ by using a relinearization matrix $\boldsymbol{B}_{u_1,u_2}$.

$\texttt{KSwitch}(\boldsymbol{m}_{E_{u_1}}, \boldsymbol{pk}_{E_{u_1}}, \boldsymbol{B}_{u_1,u_2}) \rightarrow \boldsymbol{m}_{E_{u_2}}$: outputs an encryption of m under the key $\boldsymbol{s}_{u_2}$; $\boldsymbol{B}_{u_1,u_2}$ is the relinearization matrix between keys $\boldsymbol{s}_{u_1}$ and $\boldsymbol{s}_{u_2}$ produced by user u_1.

As the degree of the evaluated polynomials defining the susceptibility functions is known beforehand, we do not need a Fully Homomorphic Encryption (FHE), and we can get a much more efficient solution by running an SHE adjusted to the needed polynomial degree. Furthermore, we leverage the relinearization function as a proxy re-encryption primitive to enable the proposed dynamic system.

4. PRIVACY-PRESERVING GENOMIC SUSCEPTIBILITY TESTING

This section revises the recently proposed privacy preserving protocol by Ayday *et al.* [2] and presents our enhanced version. Table 1 summarizes the notation for this section.

Γ^P	Set of positions of real SNPs of patient $\mathcal{P}$
γ^P	Set of positions of potential SNPs of patient $\mathcal{P}$
$\mathrm{SNP}^{P,i}$	i-th SNP for patient $\mathcal{P}$. $\mathrm{SNP}^{P,i}$ equals 0 when it belongs to γ^P, and 1 when the patient presents a variant (it belongs to Γ^P)
Ω_x	Set of positions of SNPs which are related to disease x.
$\mathrm{pr}_b^{x,i}$	$\Pr(x \mid SNP^{P,i} = b)$, with $b \in 0, 1$. Probability of developing disease x conditioned on the value of the i-th SNP
$\mathrm{c}^{x,i}$	Contribution (likelihood) of the i-th SNP $SNP^{P,i}$ to the susceptibility to disease x
$\mathrm{S}^{P,x}$	Predicted susceptibility of patient $\mathcal{P}$ to disease x

Table 1: Used Notation

There are several ways to calculate the genetic susceptibility to a given disease, but we exemplify here the weighted average method, generalizing the descriptions given in [1, 2]. Due to prior studies on a given population, a medical center $\mathcal{MC}$ knows that disease x is related to SNPs in a set Ω_x, in such a way that their presence or absence contributes by increasing or decreasing the susceptibility by a determined value (see Table 1). For this set of SNPs, the susceptibility of a patient for developing a disease x can be calculated as:

$$S^{P,x} = \frac{1}{\sum_{i\in\Omega_x} c^{x,i}} \times \left\{ \Sigma_{i\in\Omega_x} c^{x,i} \{ \frac{pr_0^{x,i}}{0-1} [\mathrm{SNP}^{P,i} - 1] + \frac{pr_1^{x,i}}{1-0} [\mathrm{SNP}^{P,i} - 0] \} \right\}. \quad (1)$$

It is possible to calculate the susceptibility by computing the Likelihood Ratio, which we omit here due to lack of space, but it is analogous to the case of Eq. (1).

4.1 Overview of Ayday *et al.*'s Scheme

Ayday *et al.* [2] propose several mechanisms with different levels of privacy to compute Eq (1) in an untrustworthy Storage and Processing Unit ($\mathcal{SPU}$), being the highest level the one using Paillier encryptions [11] to conceal the values and positions of the patient's SNPs, and a partial decryption step (which they denote proxy encryption) to interactively perform part of the process between the medical center $\mathcal{MC}$ and the $\mathcal{SPU}$. All parties are assumed semi-honest and follow the established protocols.

Patient $\mathcal{P}$ owns a biological sample and a pair of Paillier keys. The $\mathcal{MC}$ has the knowledge of the parameters for calculating the susceptibility to disease x. Additionally, [2] introduces a trusted certificate institute $\mathcal{CI}$, which performs the sequencing and encryption of the patient's DNA from her sample; it also generates the Paillier keys and shares a (regular non-homomorphic) symmetric encryption key with patient $\mathcal{P}$. The target of a privacy-preserving protocol in this scenario is to securely calculate Eq. (1) while concealing the patient DNA from both the $\mathcal{MC}$ and $\mathcal{SPU}$, outsourcing most of the processing load to the $\mathcal{SPU}$.

Prior to the protocol execution, the $\mathcal{CI}$ obtains the Paillier and symmetric encryption parameters $pp = (pp_{SE}, pp_{Pa})$ and generates and distributes the corresponding key pairs (sk_P, pk_P) for the patients. It also distributes the symmetric keys for the patient $k = (sk_{P,CI})$.

Sequencing and generation of input encryptions After $\mathcal{P}$ sends her biological sample to $\mathcal{CI}$ for sequencing, $\mathcal{CI}$ obtains the positions of SNPs in sets Γ^P and γ^P, encoded as $\mathrm{SNP}^{P,i}$. With the values of these positions, $\mathcal{CI}$ builds the Bloom Filter BF representing the positions included in Γ^P, encrypts these positions l_i and an arbitrary value l_0 (different from all the real or potential SNP indices) representing the 0 location with the patient's symmetric key, obtaining $\{l_{i,E_{P,CI}}\}$ and $l_{0,E_{P,CI}}$; it also encrypts the values of $\mathrm{SNP}^{P,i}$ using the patient's Paillier key pk_P. Then, $\mathcal{CI}$ sends the BF and $l_{0,E_{P,CI}}$ to the patient, and the pairs of encrypted positions and encrypted SNPs to the $\mathcal{SPU}$; the $\mathcal{SPU}$ can therefore index the encrypted SNPs with the encrypted position, without knowing neither the value of the SNP nor its actual position.

Susceptibility test

Step 1: In order to run the analysis, $\mathcal{MC}$ marks the location of those SNPs in Ω_x it needs to run the test and sends these positions to patient $\mathcal{P}$.

Step 2: $\mathcal{P}$ runs the BF for these positions; for those in the BF (present variants), $\mathcal{P}$ encrypts the corresponding location $l_{i,E_{P,CI}}$ and sends it to $\mathcal{SPU}$. For those not in BF, $\mathcal{P}$ sends the encryption $l_{0,E_{P,CI}}$. Additionally, she sends one half of her secret key, $sk_{P(1)}$ to the $\mathcal{SPU}$ and the other half $sk_{P(2)}$ to $\mathcal{MC}$ to perform a partial (proxy) decryption.

Step 3: The $\mathcal{SPU}$ re-randomizes the corresponding SNP encryptions $\mathrm{SNP}^{P,i}_{E_P}$ for the indices $l_{i,E_{P,CI}}$ and sends the results to $\mathcal{MC}$.

Step 4: The $\mathcal{MC}$ computes the susceptibility test Eq. (1) on the patient's encrypted SNPs by using the homomorphic properties of Paillier encryption scheme, knowing the values of $pr^{x,i}$ and $c^{x,i}$, and sends the encrypted results to the $\mathcal{SPU}$.

Step 5: The $\mathcal{SPU}$, by using its half of $\mathcal{P}$ secret key $sk_{P(1)}$, partially decrypts the result and sends it back to the $\mathcal{MC}$.

Step 6: The $\mathcal{MC}$ uses $sk_{p(2)}$ to decrypt the message and recover the required test result, $S^{P,x}$.

The security of this scheme is based on the one wayness and semantic security of the underlying Paillier encryption scheme.

4.2 Proposed Scheme

Ayday's scheme presents a series of limitations and drawbacks: firstly, it employs Paillier, an additively homomorphic scheme which translates clear-text additions into multiplications of ciphertexts and clear-text products by known values into exponentiations under encryption. Therefore, in order to keep the susceptibility parameters hidden from the $\mathcal{SPU}$, the protocol must incur in two additional communication rounds between the $\mathcal{MC}$ and the $\mathcal{SPU}$ (steps 2, 3 and 4), and it must move the bulk of the computation to the $\mathcal{MC}$ instead of the $\mathcal{SPU}$. These two facts go against the initial targets of a privacy-preserving outsourced scheme; we overcome them by relying on a somewhat homomorphic lattice-based encryption (see Section 3.3) instead of Paillier. Secondly, they use a partial decryption in order to transfer the values between the $\mathcal{MC}$ and the $\mathcal{SPU}$, in such a way that the patient is required to send part of her key to these two parties; instead, we leverage the relinearization (key switching) process of leveled cryptosystems to provide proxy re-encryption functionalities and produce the test results encrypted under the key of the $\mathcal{MC}$.

We develop our modified protocol in the same scenario, with the same parties $\mathcal{P}$, a $\mathcal{CI}$, one $\mathcal{MC}$ and one $\mathcal{SPU}$. The setup step in the $\mathcal{CI}$ is essentially unaffected: it runs the set up and key generation algorithms (in our case, for a lattice-based SHE scheme, see Section 3.3) and distributes them among the parties. It also sequences the DNA sample of the patient, builds a Bloom Filter and sends the corresponding encryptions of the real and potential SNPs to the $\mathcal{SPU}$. The main difference lies in the encrypted execution of Eq. (1) for which our protocol is able to run the whole function homomorphically with both patient data and susceptibility parameters encrypted, sending the results directly to the $\mathcal{MC}$ under its own key. Regarding the attained privacy, we achieve that the $\mathcal{SPU}$ does not get to know any data about the patient or about the susceptibility parameters.

The complete instantiation of our proposed privacy-preserving susceptibility test based on SHE is depicted in Fig. 2, and described as follows:

Step s1: $\texttt{Setup}(1^\lambda) \rightarrow pp$: The $\mathcal{CI}$ runs the setup algorithms for the SHE $\texttt{SHESetup}(1^\lambda) \rightarrow pp_{SW} = (n, q, \chi)$, and the symmetric encryptions $\texttt{SESetup}(1^k) \rightarrow pp_{SE}$, obtaining $pp = (pp_{SE}, pp_{SW})$.

Step s2: $\texttt{Gen}(pp) \rightarrow k$: $\mathcal{CI}$ runs $\texttt{SHGen} \rightarrow (\boldsymbol{sk}_u, \boldsymbol{pk}_u)$ to obtain the keys of the SHE scheme for the patients, medical center and $\mathcal{SPU}$. It also runs $\texttt{SEGen}\rightarrow sk_{P,CI}$ to generate symmetric keys between $\mathcal{P}$ and $\mathcal{MC}$.

Sequencing and generation of input encryptions

Step e1: $\mathcal{P}$ sends her biological sample to the $\mathcal{CI}$.

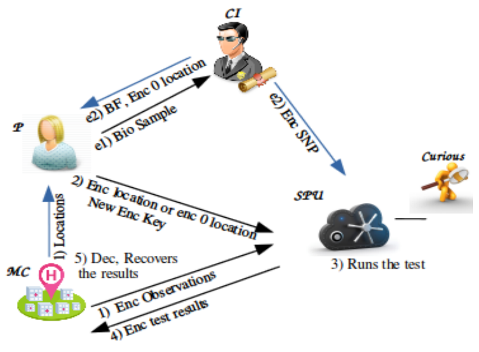

Figure 2: Proposed Privacy Preserving Scheme

Step e2: The $\mathcal{CI}$ sequences the sample, builds the patient's Bloom filter BF and symmetrically encrypts the positions $\{l_{i,E_{P,CI}}\}$ and $l_{0,E_{P,CI}}$ as in the previous scheme. Moreover, it encrypts the values of $\text{SNP}^{P,i}$ using the patient's SHE public key $\boldsymbol{pk}_P$. Then, the $\mathcal{CI}$ sends the BF and $l_{0,E_{P,CI}}$ to $\mathcal{P}$, and the pairs of encrypted positions and SNPs to the $\mathcal{SPU}$. Finally, either $\mathcal{MC}$ or $\mathcal{P}$ generates the relinearization matrix $\boldsymbol{B}_{P,MC}$ which enables the proxy re-encryption between the patient's SHE key $\boldsymbol{sk}_P$ and the $\mathcal{MC}$ key through a `KSwitch` operation, and sends it to the $\mathcal{SPU}$.

Encrypted susceptibility test

Step 1: The $\mathcal{MC}$ marks the location of SNPs in Ω_x and sends them to $\mathcal{P}$. Additionally, it sends the contributions of these SNPs to the disease x encrypted under $\mathcal{P}$'s SHE key to $\mathcal{SPU}$: $\{\boldsymbol{pr}^{x,i}_{E_P}, \boldsymbol{c}^{x,i}_{E_P}\}_{i\in\Omega_x}$.

Step 2: $\mathcal{P}$ runs the BF for these positions; for those in the BF (present variants), $\mathcal{P}$ encrypts the corresponding location $l_{i,E_{P,CI}}$ and sends it to $\mathcal{SPU}$, for those not in BF, $\mathcal{P}$ sends the encryption $l_{0,E_{P,CI}}$.

Step 3: The $\mathcal{SPU}$ computes the susceptibility test Eq. (1) on patient's encrypted SNPs and $\mathcal{MC}$'s encrypted susceptibility parameters for x by using the homomorphic properties of the SHE scheme, obtaining the encrypted value of $\boldsymbol{S}^{P,x}_{E_P}$ under $\mathcal{P}$'s key.

Step 4: The $\mathcal{SPU}$ runs $\texttt{KSwitch}(\boldsymbol{S}^{P,x}_{E_P}, \boldsymbol{pk}_P, \boldsymbol{B}_{P,MC}) \rightarrow \boldsymbol{S}^{P,x}_{E_{MC}}$ as a proxy re-encryption to get the result encrypted under the medical center's key, and sends it to $\mathcal{MC}$.

Step 5: The $\mathcal{MC}$ decrypts the clear-text test result $S^{P,x}$ of patient $\mathcal{P}$ for the disease x using its own SHE secret key.

4.3 Discussion and implementation details

The first enhancement of our modified scheme resides in replacing additively homomorphic Paillier encryptions by a lattice-based somewhat homomorphic encryption based on RLWE; this allows to perform the whole computation of the encrypted susceptibility function at the $\mathcal{SPU}$, removing step 3 in the original protocol and moving the computation in the original step 4 to the $\mathcal{SPU}$ instead of the $\mathcal{MC}$.

We achieve this goal by taking advantage of working with both SNPs and susceptibility parameters encrypted at the same time by means of the proposed SHE scheme, hence preserving both the privacy of $\mathcal{P}$'s genomic data and the confidentiality of the $\mathcal{MC}$'s test from the $\mathcal{SPU}$. Therefore, we are complying with the requirement of a truly outsourced secure computation, and the $\mathcal{SPU}$ is not only a helper party as in [2], but it can be instantiated by an untrustworthy computing infrastructure like a Cloud service, while minimizing the computation requirements on the side of the $\mathcal{MC}$ and patient $\mathcal{P}$, whose involvement in our protocol are reduced to only encrypting inputs and decrypting outputs.

Secondly, we use key switching as a proxy re-encryption, so that the patient does not need to intervene in the last step of the protocol, therefore substituting steps 5 and 6 of the original scheme, and removing the need for partial decryption keys to allow the $\mathcal{MC}$ to decrypt the final result.

Another consideration has to do with the efficiency of the computation outsourced to the $\mathcal{SPU}$. While the encryptions for the SHE-based scheme are bigger than Paillier's, they do not involve any costly exponentiation, as they translate clear-text additions into encrypted additions and clear-text multiplications into encrypted polynomial multiplications, so their performance is not far away from Paillier's. Additionally, the chosen SHE scheme allows for batching SIMD (Single Instruction Multiple Data) operations, by taking advantage of encoding the inputs in a transform domain, in such a way that one encryption can hold a vector of SNP positions or susceptibility parameters, and all the scalar products can be performed "in parallel" just as one encrypted polynomial product. This technique greatly enhances the performance of the encrypted operations and allows to reduce also the cipher expansion of the encryptions, as we will show in an extended version of this short paper.

As a final remark, we could define an alternative protocol by restricting that all input values from the $\mathcal{MC}$ are encrypted with its own key, i.e., $\mathcal{MC}$ encrypts the susceptibility parameters $pr^{P,x}_{E_{MC}}$ and $c^{P,x}_{E_{MC}}$ under its key; then, the $\mathcal{SPU}$ would proxy re-encrypt the input SNPs from the patient to the $\mathcal{MC}$'s key with `KSwitch` operations before step 3, and perform the calculation of $\boldsymbol{S}^{P,x}_{E_{MC}}$ under $\mathcal{MC}$'s key. This modification increases the computation load on the server, but it avoids that the $\mathcal{MC}$ give away its confidential values under another party's key, in case this is a restriction imposed in a practical scenario.

5. DYNAMIC PRIVACY-PRESERVING SCHEME

An additional limitation of Ayday's scheme deals with permissions and access control, in such a way that once the $\mathcal{CI}$ sends the encrypted values to the $\mathcal{SPU}$, the access control is hardwired to the used keys; and the patient provides the $\mathcal{MC}$ with part of $\mathcal{P}$'s key; in case a new $\mathcal{MC}$ joins the system, the patient has to send it the partial key; if it leaves the system, the distributed key allows the $\mathcal{MC}$ to retrieve and decrypt any partial result output by the $\mathcal{SPU}$.

Hence, it is not possible to update this access control if new $\mathcal{MC}$s join or to revoke access to results if they leave the system. These operations can be managed with the aid of the proposed proxy re-encryption, relying on the `KSwitch` primitive of the SHE, but we also sketch here a more advanced modification with a more fine-grained control over the patient consent to a set of medical centers or some susceptibility tests. For this purpose, we propose to use Attribute-Based Somewhat Homomorphic Encryption.

5.1 Attribute-Based SHE

While there are not many proposals of Attribute Based Somewhat Homomorphic Encryption (ABSHE) in the literature, a lattice-based ABSHE can be constructed by applying Gentry's homomorphic compiler [7] to a regular attribute based encryption (ABE) scheme, resulting in a cryptosystem with both properties. In order to use the ABSHE, a trusted party (our $\mathcal{CI}$) would run a setup algorithm to produce a master secret-public key pair, and a key generation algorithm to generate each of the secret keys for the patients and the $\mathcal{MC}$s. By relying on attributes, we allow to directly retrieve the public keys of the involved parties, and we also let the patient define which subset of attributes must be fulfilled by a medical center to obtain the test results.

Therefore, instead of relying on proxy re-encryption, the patient can define an access policy as a set of attributes, and the $\mathcal{CI}$ will encrypt the SNPs with the key corresponding to the allowed attribute subset instead of the patient's key. If a new $\mathcal{MC}$ joins the system, it will declare and authorize a set of attributes, and if they fulfill the patient's policy, the new $\mathcal{MC}$ can get authorization to run the tests without the intervention of the patient or the $\mathcal{CI}$. It must be noted that attributes can define not only the identity of the $\mathcal{MC}$ but also the tests, in such a way that different keys would be used for different centers and different susceptibility tests.

The most important feature of an ABSHE scheme is the ability to dynamically deal with more than one medical center which might be responsible for various tasks or tests, based on their access structure defined as a function of the medical center's attribute set. The achievable privacy levels regarding the protection against the $\mathcal{SPU}$ are the same as for the proposed scheme in Section 4.2, as all the homomorphic operations involved in step 3 of our protocol would be preserved by the ABSHE construction. An additional advantage is that it is not necessary to manage re-encryption keys, the patient does not have to generate them, and the encryptions do not have to be resent after a party joins or leaves the system.

6. CONCLUSIONS

We improved on previous privacy-preserving genetic susceptibility testing protocols by relying on a somewhat homomorphic lattice-based encryption, and using key-switching algorithms as a proxy re-encryption primitives. We allow for truly outsourced secure testing by leveraging a Ring Learning with Errors-based encryption, by achieving the same privacy and confidentiality levels as prior schemes with a better communication and round efficiency; furthermore, we move most of the computational load to an outsourced untrustworthy party, reducing the computational needs of the patients and medical centers to just encrypting the inputs and decrypting the results.

Additionally, we sketch a method with Attribute-Based Somewhat Homomorphic Encryption, which allows for a fine-grained dynamic access control by mapping the patient consent to an attribute-based policy defining the centers and the susceptibility tests that can be carried on her DNA samples. Implementation results and performance figures will be provided in an extended version of this short paper.

7. ACKNOWLEDGMENTS

This work was partially funded by the Spanish Ministry of Economy and Competitiveness and the European Regional Development Fund (ERDF) under projects COMPASS (TEC2013-47020-C2-1-R) and COMONSENS (TEC2015-69648-REDC), by the Galician Regional Government and ERDF under projects "Consolidation of Research Units" (GRC2013/009), and AtlantTIC, and by the EU H2020 Framework Programme under project WITDOM (project no. 644371)

8. REFERENCES

[1] E. Ayday, J. L. Raisaro, U. Hengartner, A. Molyneaux, and J.-P. Hubaux. Privacy-preserving processing of raw genomic data. In *Data Privacy Management and Autonomous Spontaneous Security*, pages 133–147. Springer, 2014.

[2] E. Ayday, J. L. Raisaro, and J.-P. Hubaux. Privacy-enhancing technologies for medical tests using genomic data. Technical report, 2012. Online: http://infoscience.epfl.ch/record/182897.

[3] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik. Countering GATTACA: Efficient and Secure Testing of Fully-sequenced Human Genomes. In *ACM CCS*, pages 691–702, New York, USA, 2011.

[4] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *3rd Innovations in Theoretical Computer Science Conference*, pages 309–325. ACM, 2012.

[5] M. Canim, M. Kantarcioglu, and B. Malin. Secure management of biomedical data with cryptographic hardware. *IEEE Trans. on Information Technology in Biomedicine*, 16(1):166–175, 2012.

[6] Y. Chen, B. Peng, X. Wang, and H. Tang. Large-scale privacy-preserving mapping of human genomic sequences on hybrid clouds. In *NDSS*, 2012.

[7] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013*, pages 75–92. Springer, 2013.

[8] S. Kathiresan, O. Melander, D. Anevski, C. Guiducci, N. P. Burtt, C. Roos, J. N. Hirschhorn, G. Berglund, B. Hedblad, L. Groop, et al. Polymorphisms associated with cholesterol and risk of cardiovascular events. *New England Journal of Medicine*, 358(12):1240–1249, 2008.

[9] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, volume 6110 of *LNCS*, pages 1–23. Springer, 2010.

[10] M. Naveed, E. Ayday, E. W. Clayton, J. Fellay, C. A. Gunter, J.-P. Hubaux, B. A. Malin, and X. Wang. Privacy in the genomic era. *ACM Computing Surveys (CSUR)*, 48(1):6, 2015.

[11] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238. Springer, 1999.

[12] S. Seshadri, A. L. Fitzpatrick, M. A. Ikram, A. L. DeStefano, V. Gudnason, M. Boada, J. C. Bis, A. V. Smith, M. M. Carrasquillo, J. C. Lambert, et al. Genome-wide analysis of genetic loci associated with alzheimer disease. *JAMA*, 303(18):1832–1840, 2010.

[13] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik. Privacy preserving error resilient DNA searching through oblivious automata. In *ACM CCS*, CCS '07, pages 519–528, New York, NY, USA, 2007. ACM.

[14] R. Wang, X. Wang, Z. Li, H. Tang, M. K. Reiter, and Z. Dong. Privacy-preserving genomic computation through program specialization. In *ACM CCS*, pages 338–347. ACM, 2009.