View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Infoscience - École polytechnique fédérale de Lausanne

Privacy-Preserving Exploration of Genetic Cohorts with i2b2 At Lausanne University Hospital

Gwangbae Choi[1,*], Jean Louis Raisaro[1,*], Sylvain Pradervand[2], Raphael Colsenet [2], Nathalie Jacquemont [2], Nicolas Rosat [2], and Jean-Pierre Hubaux [1]
[1] **EPFL,** [2] **CHUV**
[*] **First co-authors**

Abstract

Re-use of patients' health records can provide tremendous benefits for clinical research. One of the first essential steps for many research studies, such as clinical trials or population health studies, is to effectively identify, from electronic health record systems, groups of well-characterized patients who meet specific inclusion and exclusion criteria. This procedure is called cohort exploration. Yet, when researchers need to compile specific cohorts of patients, privacy issues represent one of the major obstacles to accessing patients' data, especially when sensitive data, such as genomic data, are involved. Because of this, cohort exploration could become extremely difficult and time-consuming. In this joint paper between the École Polytechnique Fédérale de Lausanne (EPFL) and the Lausanne University Hospital (CHUV), we address the challenge of designing and deploying an efficient privacy-preserving explorer for genetic cohorts. Our solution is built on top of i2b2 (informatics for integrating biology and the bedside), the state-of-the-art open-source framework for cohort exploration, and exploits on cutting-edge privacy-enhancing technologies (PETs) such as homomorphic encryption and differential privacy. To the best of our knowledge, our proposed solution is the first of its kind to be successfully deployed in a real operational environment within a hospital. Especially, it has been tested as one of the services of the clinical research data-warehouse of CHUV. Solutions involving homomorphic encryption are often believed to be costly and still immature for use in operational environments. In this paper, we prove the opposite by describing how actually, for specific use cases, this kind of PETs can be very efficient enablers.

1 Introduction

Secondary use of electronic health records and omics data is critical for accelerating biomarker identification for clinical development and translational medicine. Yet, as nowadays most medical institutions have clinical and omics data stored across a variety of systems, getting the information that is needed into the hands of researchers often requires substantial time and resources.

Recently, many advanced medical institutions have taken important steps to address this problem. For example, since 2015, the Lausanne University Hospital (CHUV) has been setting up a service to support clinical research[1] that is in charge of building a centralized, secure and comprehensive research data-warehouse (DWH)[a] for storing all the diverse clinical and molecular data. The goal is to provide its own researchers with an easy and cost-effective way to access the vast amount of data necessary for identifying new predictive biomarkers and rapidly finding subjects with similar clinical and omics characteristics.

Yet, because of the inherent sensitivity of these types of data, ensuring their security and privacy is a fundamental requirement for the success of such initiatives. Reinforcing the feeling of trust between patients and healthcare providers is key for fully realizing the benefits of personalized medicine.

For this reason, in the last few years, researchers from both the computer science and medical fields have started collaborating to design new solutions that protect individuals' medical and genomic privacy, in particular[2,3]. Actually, because of their identifying nature, it is in the area of genomic data that privacy issues are the most acute. However, to obtain acknowledgment and to be adopted in the real world, these solutions need to be deployed and assessed in concrete operational scenarios.

In this paper, we describe how we addressed this challenge. In particular, we designed, developed and, for the first time – to the best of our knowledge – successfully deployed in a real operational environment for clinical research a privacy-preserving solution based on cutting-edge privacy-enhancing technologies (PETs). The proposed solution

[a] A research data-warehouse is a repository that integrates de-identified information on patients from multiple sources. These can include electronic health records, lab results, genetic and research data, as well as demographic information.

makes use of differential privacy and homomorphic encryption (so far believed unpractical) to protect patients' medical and genomic privacy. This work is the result of a joint effort between the École Polytechnique Fédérale de Lausanne (EPFL) and the Lausanne University Hospital (CHUV).

For a first attempt of real deployment in a clinical research operational setting, we decided to focus on the specific use case of privacy-preserving exploratory analyses of genetic cohorts. Exploratory analyses represent a first and important step for many types of research studies. They enable researchers to quickly generate new research hypotheses and to effectively identify large groups of well-characterized patients for clinical trials or population health studies. Yet, very often these types of analyses are heavily hindered by privacy and ethical regulations within the institution itself that make it difficult, if not impossible, to access patients' private and sensitive information such as genomic data. Our solution enables researchers at a medical institution (CHUV, in our case) to efficiently and securely explore large cohorts of patients without comprising patients' privacy.

In particular, we built our solution on top of the i2b2 (Informatics for Integrating Biology and the Bedside)[4] framework that represents the state-of-the-art open-source system for exploring clinical research data-warehouses. The i2b2 framework was jointly developed, by the Harvard Medical School and MIT, to enable clinical researchers to use existing clinical data and IRB-approved genomic data for discovery research and design of target therapies. It is already well widespread in the US, as more than 80 universities and hospitals are currently using it for translational research or academic purposes[5]. Yet, as other cohort explorers, i2b2 does not provide any convincing protection for genomic data other than standard access control and data de-identification, both proven to be ineffective[6,7,8,9,10,11,12]. In our solution, patients' genetic data are homomorphically encrypted and stored in a centralized i2b2 server along with pseudonymized and de-identified clinical data. These data can be queried in a privacy-preserving way (i.e., without ever decrypting the original genomic data) by researchers willing to determine the aggregate total number of patients (or other summary statistics such as allele frequency) who meet a given set of inclusion and exclusion criteria (currently demographics, diagnoses, medications, genomics or laboratory values). According to their different access rights, researchers can receive either slightly perturbed (with noise satisfying the notion of differential privacy) – but still useful – or unperturbed query results.

We extensively tested the performance of our solution in a real operational setting on real data for different cohort sizes, and we found the overhead introduced by privacy-preserving techniques to be totally acceptable. The response time is linear in the number of selected patients and always in the order of a few seconds for simple types of realistic queries and cohort sizes.

The rest of this paper is organized as follows: In the next section, we summarize the related work. In Section 3, we briefly describe the i2b2 frameworks and the main concepts in genomics and cryptography used throughout the paper. In Section 4, we introduce the system and threat models and describe the proposed solution in detail. In Section 5, we describe its implementation and evaluate the performance using real genomic in the operational setting of CHUV. In Section 6, we discuss our findings and we conclude the paper. In the Appendix, we discuss a benchmark of some state-of-the-art homomorphic cryptosystems and we describe in detail some privacy-preserving algorithm used in the proposed solution.

2 Related Work

For many years, researchers have assumed that releasing anonymized genomic data for research purposes would not compromise participants' privacy. However, it has been shown at multiple rounds[6,7,8,9,10,11,12] that standard anonymization techniques are ineffective on genomic data. As a response to this concerns, many solutions were proposed in the last few years, with the goal of protecting genomic data and enabling its utility for medical research. We can put them in two main categories: (i) approaches that focus on the protection of data confidentiality against illegitimate access and (ii) approaches that mitigate the risk of sensitive attribute inference from legitimate access genomic data.

The first category includes works such as the one by Canim et al.[13] where the authors describe how to outsource investigations on genomic data to a commercially available cryptographic hardware. Also in this category, a more recent study by Kamm and co-authors[14] proposes a new scheme for generating aggregated statistics on genomic data by using secure multi-party computation based on homomorphic secret sharing. Their technique requires the presence of multiple non-colluding servers. Xie *et al.* introduce in their work[15] a novel cryptographic strategy based on secure multi-party computation to securely perform meta-analysis for genetic association studies in large consortia, whereas Wang *et al.*[16] also rely on secure multi-party computation techniques to securely compare genomes across institutions.

Finally, several other works [17,18,19,20,21] propose using homomorphic encryption to protect genomic information in order to allow researchers to perform some statistics directly on the encrypted data and decrypt only the final result.

The second category includes the work by Uhler *et al.* [22] that proposes new methods for releasing aggregate results from genome-wide association studies (GWAS) without compromising a participant's privacy, by focusing on the differentially private release of minor allele frequencies. A similar approach is also adopted by Johnson and Shmatikov [23], Yu et al. [24] and Simmons and Berger [25,26,27], whereas Tramer *et al.* [28] investigated a relaxation of differential privacy thus providing a better tradeoff between privacy and utility.

In our work, differently from those mentioned above that address a single problem, we mitigate both risks. Our solution makes use of both homomorphic encryption (HE) to protect patients' genomic data confidentiality against illegitimate access at rest and during processing and differential privacy (DP) to protect against re-identification attacks. Whereas for the latter we propose a straightforward application of DP, we designed a new optimized solution based on HE that outperforms the state-of-the-art. Moreover, to the best of our knowledge, our solution is the first to be deployed and tested in a real medical research environment. Another work by McLaren and co-authors [29] describes a successful deployment in a medical operational environment. Yet, in that work homomorphic encryption is used to protect data for personalized medicine and not for research purposes.

Two research contributions from the i2b2 community describe how to use genomic data within the i2b2 framework. Phillips *et al.* [30] propose to use a genomic ontology directly in the i2b2 native framework, whereas Gabetta et al. propose BigQ [31], an i2b2 plugin handling genomic variants and their annotations. Yet, both works do not at all address privacy problems due to the use of genomic information. They simply rely on the i2b2 standard security protections that are based on simple access control mechanisms as described in the work by Murphy et al. [32]. To the best of our knowledge, our work is the first to combine i2b2 with advanced privacy-enhancing technologies for genomic data protection.

3 Preliminaries

In this section, we briefly summarize the main concepts in cryptography and genomics that are used in the paper.

3.1 Notation

In this paper, we denote the cardinality of a set A by $|A|$. We denote x uniformly chosen from the set X as $x \xleftarrow{U} X$.

3.2 Cryptographic Background

In this section, we briefly explain the cryptographic concepts necessary for understanding the rest of the paper.

Homomorphic Encryption. Homomorphic encryption (HE) is a special type of encryption that supports computation on encrypted data. In 2009, Craig Gentry [33] introduced for the first time a special type of HE enabling for *arbitrary computations* on ciphertexts called fully homomorphic encryption (FHE). More formally, FHE could be described as follows.

Let $\mathsf{CS}(K_p, K_s, P, C, \mathcal{E}, \mathcal{D})$ be a cryptosystem with the public key space K_p, the secret key space K_s, the plaintext space P, the ciphertext space C, the encryption function $\mathcal{E} : P \times K_p \rightarrow C$, and the decryption function $\mathcal{D} : C \times K_s \rightarrow P$. We say that the cryptosystem CS is *fully homomorphic* if and only if for any function $f : P \times P \rightarrow P$ in the plaintext domain, there exists another function $h : C \times C \rightarrow C$ in the ciphertext domain such that

$$\mathcal{D}(h(\mathcal{E}(m_1, k_p), \mathcal{E}(m_2, k_p)), k_s) = f(m_1, m_2) \tag{1}$$

for any $m_1, m_2 \in P$, $k_p \in K_p$ and $k_s \in K_s$.

Yet, despite its complete functionality, FHE is totally unpractical as it introduces significant computational and storage overheads that make it unusable for real-world applications. For this reason, many variations of FHE have been proposed in the last few years with the goal of moving towards better efficiency by sacrificing some flexibility. Such cryptosystems are called *practical* homomorphic cryptosystems, and according to their functionality, they can be clas-

sified as additively homomorphic if they only satisfy addition of of ciphertexts, multiplicatively homomorphic if they only satisfy multiplication, or *leveled* (or *somewhat*) homomorphic if they support both addition and multiplication but a limited number of multiplications.

Our proposed solution is based on the Fan and Vercauteren (FV) cryptosystem[34] which is a lattice-based leveled homomorphic encryption scheme based on the *Ring Learning With Errors* (RLWE) problem. The FV scheme ensures indistinguishability against chosen plaintext attack if the standard RLWE problem is hard. Moreover, as other lattice-based cryptosystems, it is supposed to be quantum-resistant. Let ℓ be a polynomial degree, q be a coefficient modulus, t be a plaintext modulus, $\mathcal{X}$ be a distribution over a polynomial ring $\mathbb{Z}_q[x]/(x^\ell+1)$, and $M \in \mathbb{Z}_t[x]/(x^\ell+1)$ be a plaintext polynomial. Let $s \xleftarrow{U} \mathbb{Z}_2[x]/(x^\ell+1)$ be the secret key and $\boldsymbol{p} = (p_0, p_1) = (-(a \cdot s + e) \bmod q, a)$, be the public key where $e \leftarrow \mathcal{X}$ and $a \xleftarrow{U} \mathbb{Z}_q[x]/(x^\ell+1)$. Then the FV scheme works as follows:

- *Encryption:* $Enc(M, \boldsymbol{p}) = \boldsymbol{c} = (c_0, c_1) = ((p_0 \cdot u + e_1 + \lfloor q/t \rfloor \cdot M) \bmod q, (p_1 \cdot u + e_2) \bmod q)$, where $u, e_1, e_2 \leftarrow \mathcal{X}$
- *Decryption:* $Dec(c, \boldsymbol{s}) = \left\lfloor \frac{t}{q} \cdot ((c_0 + c_1 \cdot s) \bmod q) \right\rceil \bmod t$
- *Homomorphic Addition:* $\boldsymbol{c} \oplus \boldsymbol{c'} = ((c_0 + c'_0) \bmod q, (c_1 + c'_1) \bmod q)$

We do not report the definition of homomorphic multiplication as it is not used in our solution. For further details we invite the reader to refer to the original paper[34]. Note that we choose the FV scheme because, to the best of our knowledge, it provides the best performance in terms of homomorphic computations and storage overhead for the operations required in the proposed solution. We compared the FV scheme with the Elliptic curve ElGamal (EC-ElGamal) cryptosystem[35] and Yet Another Somewhat Homomorphic Encryption (YASHE) scheme[36]. Benchmark details can be found in the Appendix.

Ciphertext Packing. Ciphertext packing[37] is a technique that can be used to reduce the overall size of the ciphertext and improve the efficiency of homomorphic operations. Despite recent advances, practical homomorphic encryption is still quite expensive. This is because security considerations require ciphertexts to be large, thus slowing down homomorphic computations. Ciphertext packing represents the main technique for dealing with this problem as a vector of plaintext values, and not a single value, can be encrypted in only one ciphertext. Homomorphic operations are applied to these vectors component-wise.

More formally, let $\mathsf{CS}(K_p, K_s, P, C, \mathcal{E}, \mathcal{D})$ be a cryptosystem, and $m_0, m_1, \cdots$ be the messages to be encrypted where $m_i \in M,\ \forall i$. Let also $n = \left\lfloor \frac{|P|}{|M|} \right\rfloor$ and $P_1, P_2, \cdots P_n$ be n independent subspaces of P where $|P_j| \geq |M|,\ \forall j$. When $|P| \geq 2 \cdot |M|$, we can encrypt at most n messages into one ciphertext by encrypting $m' = m_1 p_1 + m_2 p_2 + \cdots + m_n p_n$, where p_j is the basis of the subspace P_j.

For example, when $P = \mathbb{Z}_q[x]/(x^\ell+1)$ and $M = \mathbb{Z}_q$, we can encrypt at most ℓ messages into one ciphertext with $m' = m_0 + m_1 x + \cdots + m_{\ell-1} x^{\ell-1}$.

Differential Privacy. Differential privacy is an approach to privacy-preserving reporting of results, introduced by Cynthia Dwork[38], that guarantees that a given statistic, $f(D) = R$, computed on a dataset D_1 behaves almost the same when computed on the neighbor dataset D_2 that differs from D_1 in exactly one element. More formally we have that

$$\Pr\left[f(D_1) = R_0\right] \leq \exp(\epsilon) \cdot \Pr\left[f(D_2) = R_0\right], \tag{2}$$

where the parameter ϵ is a privacy parameter: the closer it is to 0 the more privacy is ensured. The most straightforward method for achieving ϵ-differential privacy consists in perturbing the output of the statistic with noise drawn from the Laplace distribution with mean 0 and scale $\frac{\Delta f}{\epsilon}$, where Δf is known as the *sensitivity* of f:

$$\Delta f = \max_{D_1, D_2} ||f(D_1) - f(D_2)||_1. \tag{3}$$

Differently from *k-anonymity*, differential privacy guarantees privacy against an adversary regardless of his prior knowledge.

3.3 Genomic Background

In this section, we briefly introduce the basic genomic concepts used in this paper.

Genomic Variant. Around 99.9% of the whole human genome is identical between any two individuals and the remaining part ($\sim$0.1%) is referred to as *genetic variation*. Most commonly, this genetic variation comes from differences in single nucleotides, called *single nucleotide variants* (SNVs). Yet, there exist many other types of genetic differences also involving multiple nucleotides such as *insertions/deletions* (INDELs), *duplications* (DUPs), *copy number variants* (CNVs) and more complex *structural variants* (SVs). In the human population, a given genetic locus can have several possible versions (or alleles) with different genetic variations. Because of the diploid nature of somatic human cells, a human genome is comprised of two sets of chromosomes – one inherited from each parent. Therefore, each individual either has two copies of the same allele/variant (*homozygous*) or two copies of different alleles/variants (*heterozygous*).

Variant Call Format (VCF). The Variant Call Format (VCF)[39] is the main format for storing genetic variants with respect to the reference genome of one or more individual genomes. The VCF consists of two parts, header and content. The header contains the meta-information about the file and data along with the definition of file variables. The content contains the information about the genetic variants for each individual. Each variant is uniquely identified by (i) its chromosomal position (CHR, POS), (ii) the reference allele (REF), and (iii) the alternate allele (ALT) We can separate the content into two parts by the characteristics of the data. Each line of the content corresponds to the information about a variant and the genotype (i.e., the value) of this variant for each individual in the VCF. We call the information about the variant, such as CHR, POS, REF, ALT, meta-data. Meta-data is not sensitive from the privacy perspective as it is public information, as opposed to genotype information that is sensitive and must be protected.

In the VCF file, a genotype is represented by two numbers separated by either '|' or '/'. When it is separated by '|', the genotype is phased (i.e, we know which of the two chromosomes holds which allele). Whereas, when it is separated by /, the genotype is unphased (i.e., there is no information on which chromosome holds which allele). Each number represents the allele value. When it is 0, it means that the allele value is equal to the reference allele. When it is 1, it means the allele value is equal to the alternate allele. When the allele has not been genotyped correctly and there is no information about its value, we put '.' instead of any number. Such an event is named *no-call*.

In our solution, we assume that the VCF file was processed in such a way that entries with multiple alternate alleles were separated in several lines, with one allele per line.

4 Methodology

In this section, we introduce CHUV's system and threat models. We then outline the functional requirements that our system should satisfy and finally we describe our proposed privacy-preserving solution in details.

4.1 System Model

We propose a solution for privacy-preserving exploration of genetic cohorts at the Lausanne University Hospital (CHUV). The CHUV's information system consists of two physically separated networks, as depicted in Fig.1, each of them hosts different services: (i) the main network of the hospital, also called *clinical network* and (ii) a *research network* that is shared with the University of Lausanne (UNIL).

- *Clinical network.* The clinical network is used for hospital's clinical daily activities. It hosts all services used for daily healthcare and administration purposes along with the clinical research data-warehouse[b] (DWH-RC) that contains pseudonymized clinical and genomic data of patients. This network is protected by a firewall that blocks all incoming network traffic.
- *Research network.* The research network is also protected by a firewall that blocks all incoming network traffic except the one from the clinical network. It hosts multiple services used by clinical or academic researchers

[b]The detailed description of the clinical network is out of the scope of this work.

in their research activities; i2b2 is one of these services. The i2b2 service is composed of (i) an i2b2 server to which pseudonymized and de-identified clinical data along with pseudonymized and encrypted genomic data are pushed from the DWH-RC and (ii) a proxy server which is devoted to support the decryption phase and the storage of partial decryption keys. Both servers are physically separated from each other, protected by different firewalls and equipped with an intrusion detection system. Moreover, the two servers cannot communicate with each other. Researchers already in the network access the i2b2 service after authentication through an internal Web-client.

The main purpose of the CHUV's IT architecture is to isolate data that is used for clinical care and that is accessible only to few trusted and authorized individuals from data used for research activities that can be accessed by several researchers through less restrictive authorization and authentication procedures. All communications are protected through encryption.

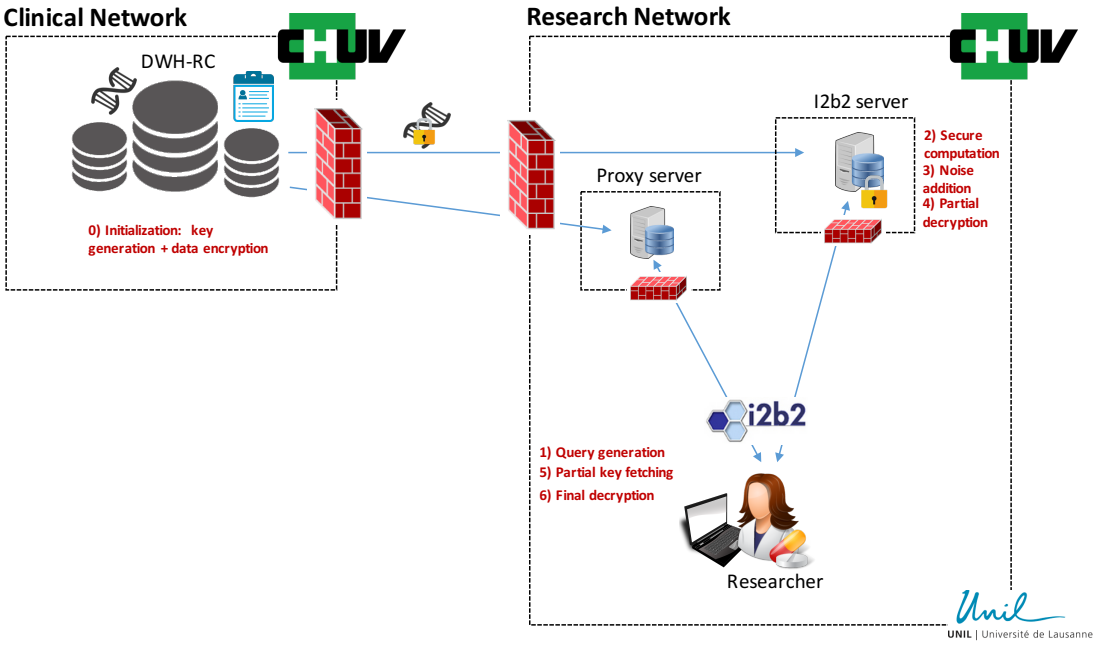


Figure 1: Architecture of the proposed solution.

4.2 Threat Model

In this project, we consider two types of potential attackers: (i) a *honest-but-curious* (or *semi-honest*) adversary at the i2b2 server or at the proxy server who honestly follows the protocol but tries to passively infer some private information about the patients, and (ii) a *malicious-but-covert* adversary that wants to re-identify a patient by performing multiple malicious, but legitimate, queries to the i2b2 service. We consider the DWH-RC as a trusted party as it is the generator and owner of the data.

- The honest-but-curious attacker can be represented by a careless or disgruntled employee of the hospital (i.e., an insider) or a hacker who has illegitimate access to the i2b2 server and tries to obtain patients' private genomic and clinical information without altering the protocol. Note that although this information was pseudonymized and there is no direct link with patients' identities, re-identification would still be possible due to the identifying nature of the genome and to some auxiliary (and often publicly available) information (e.g., public genomic databases, recreational websites, online social networks, etc.) that the attacker might exploit[6,7,8,9,10,11,12]. As a consequence, a potential loss of clinical and genomic data could be extremely dangerous, not only for the patients but also for the reputation of the medical institution itself. Re-identified health-care records are nowadays extremely valuable for hackers as, according to a recent report by IBM[40], their value on the black market is as

much as 60 times more that of stolen credit cards. We assume that the i2b2 server and the proxy server do not collude or, in other words, that they are not compromised simultaneously.

- The malicious-but-covert adversary can be represented by a malicious but legitimate user of i2b2 (e.g., a malicious researcher) or hacker who breaks into the research network and uses the i2b2 service to re-identify an individual in a subset of patients with specific clinical characteristics. In particular, an attacker with already some genomic information about the victim (e.g., the value of some of her genetic variants) might repeatedly query the i2b2 service with this genomic information and use the system as an oracle. As such, he could re-identify the presence of the victim in a sensitive subset of individuals (e.g., all cancer patients or all HIV-positive patients, etc.) and infer her health status. For example, the attacker could exploit the aggregate information obtained from the cohort explorer as described in well-known attacks such as Homer's attack[6] and the Beacon attack[7].

Therefore, with these adversarial models, there are two potential privacy threats that we need to address with our proposed solution: (i) loss of patients' health data confidentiality due to illegitimate data access and (ii) patients' re-identification and resulting sensitive attribute disclosure from legitimate data access. Data confidentiality can be protected at rest and during processing by using homomorphic encryption, whereas the re-identification risk can be mitigated by perturbing the query result in order to satisfy the notion of differential privacy.

4.3 Functional Requirements

As a first example of advanced privacy-preserving solution for cohort exploration in a medical operational setting, we decided to focus on a set of simple functional requirements. These requirements are based on the functionalities of other well-known tools for exploration of genetic cohorts such as the Beacon system of Global Alliance for Genomics & Health (GA4GH)[41] and the ExAC browser of the Broad Institute[42]. For example, through the Beacon system researchers can query a database of genomes for the presence of a specific mutation, whereas researchers using the ExAC browser can also have information about the alternate allele count and frequency of the queried mutation.

In particular, a user of our system should be able to obtain for all genetic variants in a selected chromosomal range:

- Reference/alternate allele frequencies
- Number of mutated genotypes (i.e., with at least one alternate allele)
- Number/frequency of genotypes that are homozygous with respect to the reference/alternate allele
- Number/frequency of genotypes that are heterozygous (with or without phase information)

4.4 Proposed Solution

Our solution for the privacy-preserving exploration of genomic cohorts consists of three main parts: (i) *system initialization* where cryptographic keys are generated and the genetic variants in the VCF file are encoded, encrypted and pushed to the i2b2 server along with de-identified clinical data for secure storage, (ii) *user assignment*, where access rights and cryptographic keys are assigned to each new user in the system and (iii) *query execution*, where the user builds a new query that is then sent to the i2b2 server and processed in a privacy-preserving way, i.e., without ever decrypting the data. The query result is then decrypted by the user via the i2b2 Web-client.

System Initialization. The system initialization phase takes place at the clinical research data-warehouse (DWH-RC) where clinical and genomic data are stored with patients' pseudonyms and can be accessed only by a group of few trusted and authorized individuals. The first step consists in setting the parameters of the FV cryptosystem (l, t and q) according to the desired security level (e.g., 80 bits security) and the maximum number of additions and multiplications to be supported by the system. In our case, the maximum number of additions should be at least twice the number of individuals in the database because it corresponds to the maximum number of alleles that could be involved in a counting query. Multiplication is not needed. For the optimal selection of the FV parameters, we invite the reader to

refer to the original work by Fan and Vercauteren[34]. Then, a public key $\boldsymbol{p}$ and a secret key $\boldsymbol{s}$ are generated as described in Section 3.2.

After key generation, the VCF file is parsed and each genotype is encoded following the scheme described either in Table 1 (for phased genotypes) or in Table 2 (for unphased genotypes). For simplicity, in the rest of the paper we describe only the encoding for unphased genotypes described in Table 2 (but the encoding in Table 1 is equally supported by the proposed solution). As the number of potential genotype values is 6 (we are also including *no-calls* as they can be used when allele or genotype frequencies are computed), we use three values for genotype encoding: the first two values indicate the presence of zero, one or two alternate alleles, whereas the third value reports the number of no-calls in the genotype.

Genotype value	Genotype encoding			
	gt1	*gt2*	*gt3*	*no_call*
.|.	0	0	0	2
.|0	0	0	0	1
.|1	1	0	0	1
0|.	0	0	1	1
1|.	0	1	0	1
0|0	0	0	0	0
0|1	1	0	0	0
1|0	0	1	0	0
1|1	0	0	1	0

Table 1: Encoding for phased genotypes

Genotype value	Genotype encoding		
	gt1	*gt2*	*no_call*
./.	0	0	2
./0	0	1	1
./1	1	0	1
0/0	0	0	0
0/1	1	0	0
1/1	0	1	0

Table 2: Encoding for unphased genotypes

For each individual in the VCF file, consecutive genotypes are packed into 3 sets of polynomials by using the packing technique described in Section 3.2. Let $(gt1_0, gt2_0, no\_call_0), (gt1_1, gt2_1, no\_call_1), (gt1_2, gt2_2, no\_call_2)$,... be encoded genotypes where $(gt1_i, gt2_i, no\_call_i)$ is the encoded genotype for variant i. Then, we can pack at most ℓ genotypes in three polynomials

$$\boldsymbol{gt1_j} = gt1_{j\ell} + gt1_{j\ell+1}x + gt1_{j\ell+2}x^2 + \cdots + gt1_{(j+1)\ell-1}x^{\ell-1}, \tag{4}$$

$$\boldsymbol{gt2_j} = gt2_{j\ell} + gt2_{j\ell+1}x + gt2_{j\ell+2}x^2 + \cdots + gt2_{(j+1)\ell-1}x^{\ell-1}, \tag{5}$$

$$\boldsymbol{no\_call_j} = no\_call_{j\ell} + no\_call_{j\ell+1}x + no\_call_{j\ell+2}x^2 + \cdots + no\_call_{(j+1)\ell-1}x^{\ell-1}, \tag{6}$$

where ℓ is the polynomial degree and j the ciphertext index. Polynomials $\boldsymbol{gt1_j}$ and $\boldsymbol{gt2_j}$ are finally encrypted with the public key $\boldsymbol{p}$ at the DWH-RC to obtain $\boldsymbol{cipher\_gt1_j}$ and $\boldsymbol{cipher\_gt2_j}$ which are pushed to the i2b2 server for storage along with $\boldsymbol{no\_call_j}$, patients' pseudonyms, de-identified clinical data, and $\boldsymbol{p}$, as shown in Fig.2. Note that $\boldsymbol{no\_call_j}$ does not need to be encrypted as it does not leak any information on the value of the genotype.

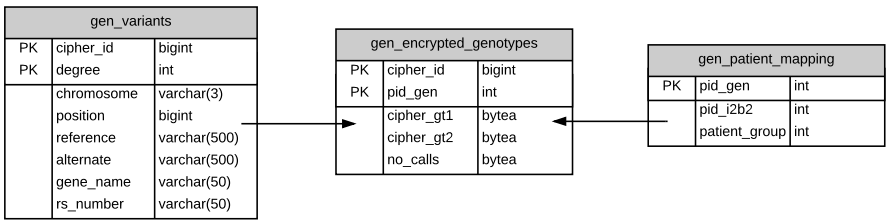


Figure 2: Database scheme for storing encrypted genetic variants at the i2b2 server.

User Assignment. Assignment of a new user also takes place at the DWH-RC. During this phase, for each new user, the secret key $\boldsymbol{s}$ is randomly divided into two shares s_1 and s_2 such that $\boldsymbol{s} = s_1 + s_2$. The two partial secret keys s_1 and s_2 are then sent, for storage, to the i2b2 server and proxy server, respectively. This procedure avoids a single point of failure in the system: if one of the two servers is compromised, data are still protected because only with the full secret key $\boldsymbol{s}$ the attacker can successfully decrypt. Note that ideally, to ensure better security, the i2b2 server and the proxy

server should be part of two completely different organizations. Yet, because of the CHUV's infrastructure these two servers are within the same network. Nevertheless, they are physically separated, managed by different administrators and cannot communicate with each other.

Access rights for each new user are also assigned during this phase and stored on the i2b2 server, as shown in Fig.3. Different users might have different rights to access patients' private information. Our system provides full customization on three different levels of access: (i) query result, (ii) patients' set, and (iii) variants' set. For example, a junior researcher might have access only to perturbed query results, where some noise has been added on the true result to satisfy the notion of differential privacy, whereas a senior researcher can obtain accurate information. Also, depending on his specialization or on the IRB-approved study protocol, a researcher might have access only to a given subset of patients or a given subset of genetic variants. For example, an oncologist might have access only to data of patients with cancer and might not be allowed to query genetic variants related to other diseases such as diabetes or coronary artery disease.

We acknowledge that if the i2b2 server is compromised by a honest-but-curious adversary, information on users' access rights could leak because they are stored in the clear. Protecting users' access rights is not the focus of this project. Yet, we are planning to address such an issue in future work by exploring even more sophisticated solutions based on *attribute-based somewhat homomorphic encryption (ABSHE)*[43].

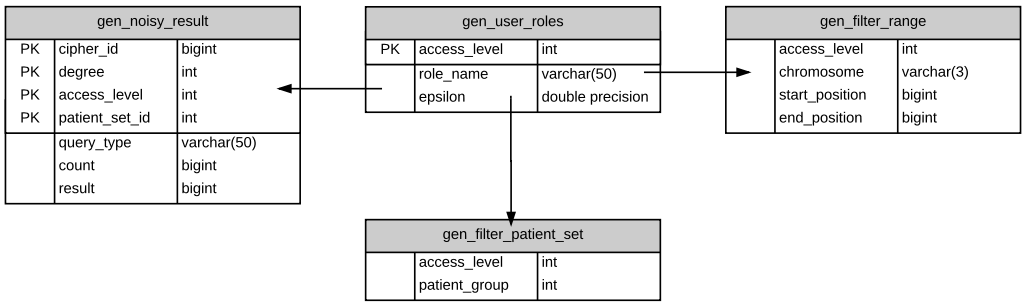


Figure 3: Database scheme for storing users' access rights at the i2b2 server.

Query Execution. The query execution consists of three phases: (i) the query generation at the user-side and (ii) the query processing at the server-side and (iii) the result decryption at the client-side.

(i) In the query generation, after password-based authentication, the user of our privacy preserving cohort explorer (i.e., CHUV's researcher) builds his query in two steps through the i2b2 Web-client. In the first step, he selects a set of inclusion and exclusion clinical criteria in order to identify the set of patients for which he wants to explore their genetic data. For example, a researcher might want some aggregate genetic information on some specific variants for people with cancer who underwent a specific treatment and who had a positive outcome. In the second step, once the patient set has been identified, the user selects the set of variants of interest and the summary statistics he wants to obtain from the ones specified in Section 4.3 and finally submits the query.

(ii) During the query processing, the i2b2 server verifies the access rights of the querier and his query definition. If the verification is successful, the server retrieves from the database the list of ciphertexts containing the encrypted genotypes of the variants specified by the query. The ciphertexts are then used to homomorphically compute the summary statistics requested by the user. We designed different secure algorithms for computing the different summary statistics. For example, the *alternate allele frequency* for a set of variants can be securely computed as described in Algorithm 1, where $\oplus$ represents homomorphic addition. For completeness, we provide in the Appendix the description of all secure algorithms needed to compute the other summary statistics outlined in Section 4.3. After computation of encrypted summary statistics, the i2b2 server partially decrypts them with the first part of the users' secret key s_1 (which is stored in the i2b2 database). In particular, from a ciphertext $c = (c_0, c_1)$ we obtain, after partial decryption, a new ciphertext $c' = (c'_0, c'_1)$, as described in

Section 4.5. Depending on the access level of the user, some noise sampled from the Laplace distribution might be homomorphically added by the i2b2 server to the result in order to satisfy the notion of *differential privacy* and prevent malicious re-identification of patients. Finally, the i2b2 server sends back to the user the encrypted polynomial c'_1 and the coefficients of the encrypted polynomial c'_0 corresponding the variants specified by the query and user's access level. Note that only the specified coefficients of the encrypted polynomial c'_0 are sent by the server to the user because we do not want him to obtain the summary statistics of all the variants packed in the same ciphertext, but only the ones he has requested and access to.

(iii) At the user-side, the Web-client fetches the second part of the user's secret key s_2 from the proxy server and performs the full decryption to obtain the final results of the query, as described in Section 4.5. For performance reasons, part of the full decryption (i.e., the polynomial multiplication $c'_1 \cdot s_2$) could be run at the proxy server. Note that, by observing only s_2 and c'_1, the proxy server cannot infer anything about the plaintext.

Algorithm 1: Secure allele frequency computation

Input : Patient set P, variant set V packed into the same ciphertext, set of the number of no calls N_p of each patient $p \in P$, and set of encrypted genotypes G_p of each patient $p \in P$. G_p consists of two ciphertexts as $G_p.cipher\_gt1$ and $G_p.cipher\_gt2$
Output: Map between the set of variants and the ciphertexts containing the result

$M \leftarrow$ an empty map // Map of key/value pairs containing encrypted results per set of variants which can be computed together
$M.addEntry(V, (\mathsf{Enc}(0), 0))$
for $p \in P$ **do**
  ones $\leftarrow$ Set of variants in V whose $N_p = 1$
  twos $\leftarrow$ Set of variants in V whose $N_p = 2$
  $M' \leftarrow$ an empty map
  for $(key, value) \in M$ **do**
    if $key \cap twos \neq \varnothing$ **then**
      $M'.addEntry(key \cap \text{twos}, value)$ // Computation for genotype ./.
    end
    if $key \cap ones \neq \varnothing$ **then**
      $M'.addEntry(key \cap \text{ones}, (value[0] \oplus G_p.cipher\_gt1, value[1] + 1))$ // Computation for genotype ./0 and ./1
    end
    if $key \setminus (ones \cup twos) \neq \varnothing$ **then**
      $M'.addEntry(key \setminus (\text{ones} \cup \text{twos}), (value[0] \oplus G_p.cipher\_gt1 \oplus 2 \cdot G_p.cipher\_gt2, value[1] + 2))$ // Computation for genotype 0/0, 0/1 and 1/1
    end
  end
  $M \leftarrow M'$
end
return M ▷ $value[0]/value[1]$ will be computed at the client side

4.5 Partial Decryption With FV Scheme

Because the Fan and Vercauteren (FV) secret key s has been split into two shares stored at the i2b2 server and at proxy server respectively, decryption has to be done in two steps. The original FV cryptosystem does not have a partial decryption algorithm. Here we described how this additional feature can be easily achieved.

Definition 1 (Partial Key Generation)**.** *Let q be the ciphertext modulus, ℓ be the polynomial degree and $s \in \mathbb{Z}_q[x]/(x^\ell + 1)$ be the secret key[34]. Then, the partial key set (s_0, s_1) can be generated as $s_0 \xleftarrow{U} \mathbb{Z}_q[x]/(x^\ell + 1)$ and $s_1 = s - s_0$.*

Definition 2 (Partial Decryption)**.** *Let q be the modulus, ℓ be the polynomial degree, t be the plaintext modulus, (s_0, s_1) be the partial key set from Definition 1 and $\mathbf{c} = (c_0, c_1)$ be the ciphertext where $c_0, c_1 \in \mathbb{Z}_q[x]/(x^\ell + 1)$. Then, we can define the partial decryption and the full decryption as follows,*

$$Partial\_Dec(\boldsymbol{c}, s_0) = \boldsymbol{c'} = (c'_0, c'_1) = (c_0 + c_1 \cdot s_0,\ c_1)$$
$$Full\_Dec(\boldsymbol{c'}, s_1) = Dec(\boldsymbol{c'}, s_1),$$

where $Full\_Dec(Partial\_Dec(\boldsymbol{c}, s_0), s_1) = Dec(\boldsymbol{c}, s)$.

Proof. We have

$$
\begin{aligned}
Full\_Dec(Partial\_Dec(\boldsymbol{c}, s_0), s_1) &= \left\lfloor \frac{t}{q}((c_0 + c_1 \cdot s_0 + c_1 \cdot s_1) \bmod q) \right\rceil \bmod t \\
&= \left\lfloor \frac{t}{q}((c_0 + c_1 \cdot s) \bmod q) \right\rceil \bmod t \\
&= Dec(\boldsymbol{c}, s)
\end{aligned} \tag{7}
$$

By Definition 1, $s_0 + s_1 = s$, so Eq.(7) holds. Thus, $Full\_Dec(Partial\_Dec(\boldsymbol{c}, s_0), s_1)$ is equivalent to $Dec(\boldsymbol{c}, s)$. □

5 Experiment and results

In this section, we describe how we implemented and deployed our solution in the real operational setting of the Lausanne University Hospital. We evaluate its performance on real genomic and clinical data.

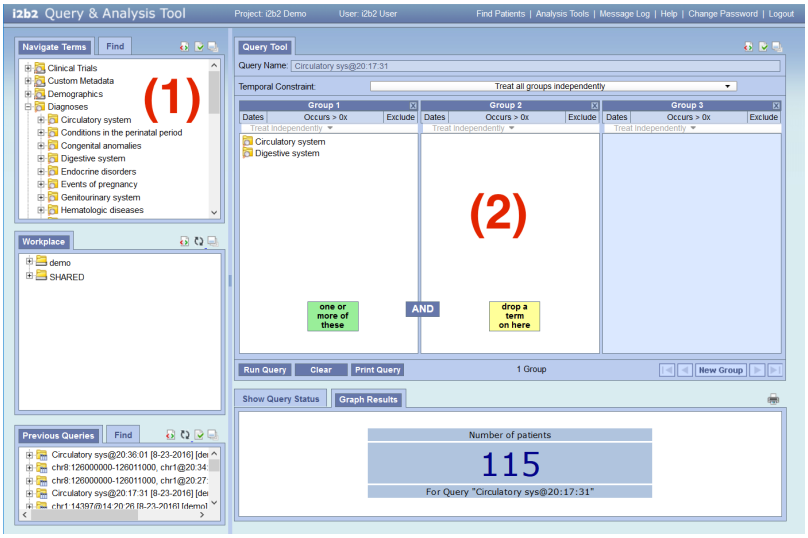


Figure 4: i2b2 native Web-client.

5.1 Plugin Implementation

We implemented our privacy-preserving solution as a plugin of the i2b2 framework. The i2b2 architecture consists of two major pieces. The first is the back-end infrastructure (the "Hive") that is responsible for the security aspects, the access rights and for managing the underlying data repository. The second piece is the user Web-client: a front-end application suite of query and mining tools that enables users to ask questions about patients' data on the i2b2 server. The native version of i2b2 does not include any support neither for privacy-preserving data processing nor for managing genomic data but it is only focused around cohort identification based mostly on clinical and demographic data. As shown in Fig.4, after logging in, a user can drag-and-drop search terms from the clinical ontology (1) into the Venn diagram-like interface (2) to construct his cohort of patients. Yet, i2b2 is easily extensible thanks to its modular design. As a consequence, we implemented our privacy-preserving plugin as a totally independent module that can be easily loaded during the setup of i2b2. Our plugin is composed of four main parts: (i) a data importation tool, (ii) a back-end module for the i2b2 server, (iii) a back-end module for the proxy server, and (iv) a front-end module for the i2b2 native Web-client.

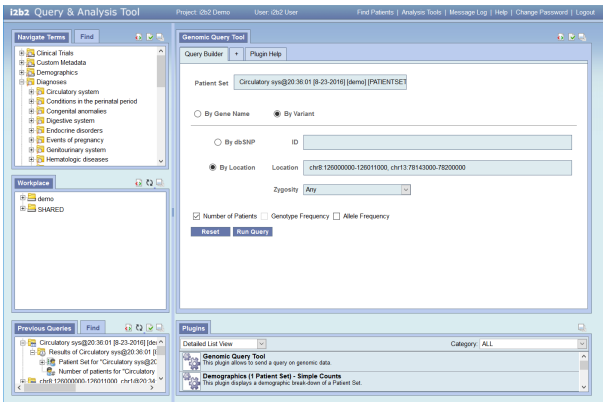
In the followings, we briefly describe each of these components.

Data Importation Tool. The importation tool is written in C++ and is responsible for the system *initialization* and the new *user assignment*. For the system initialization, it takes as input the VCF file, the access rights policies and the FV public and secret keys. The tool parses the VCF file, encodes and packs the genotypes into polynomials and encrypts them by using the FV public key. The final output is a SQL script that can be used to import data in the i2b2 SQL database. For the new user assignment, the importation tool takes as input the new user's identifier, his access level and the FV secret key. As output, it generates a SQL script to import into the i2b2 server the new user's access level along with the first part of the secret key and into the proxy server the second part of the secret key.

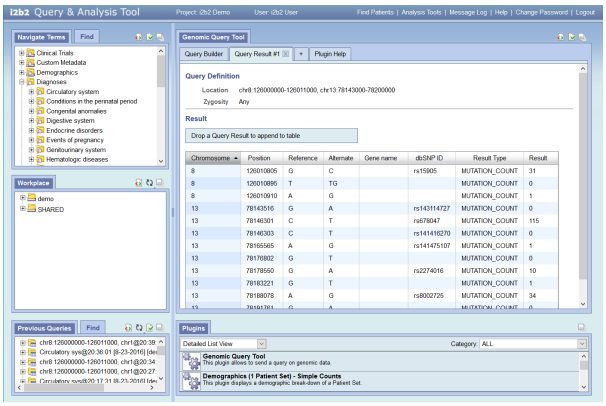
i2b2 Server Module. The i2b2 server back-end module consists of two parts: the "main cell" and the "crypto engine". The main cell is written in Java and is part of the i2b2 server application. It is responsible for managing the data repository, handling the queries, computing homomorphic addition of ciphertexts, and, according to the user's access rights, adding noise on the computation result to satisfy the notion of differential privacy. The crypto engine is written in C++ and it is used for the partial decryption at server side. The Java Native Interface (JNI) is used to call function in the C++ crypto engine from the main cell.

Proxy Server Module. The proxy server back-end module has a similar structure to the i2b2 server module. The main cell, written in Java, is responsible for managing users' partial keys stored in the data repository, whereas the crypto engine is responsible for helping the user with the full decryption by performing polynomial multiplication.

Web-client Module. The Web-client front-end module is written in JavaScript and can be loaded from the native i2b2 Web-client. It consists of a *query builder* (Fig.5a), where the user can drag-and-drop a patient set (previously constructed through the native interface) and type into a search bar a set of genetic variants of interest, and a *result visualizer* (Fig.5b), where the user can visualize the results of his current and previous queries. The user is allowed to enter genetic variants by gene name, dbSNP identifier or chromosomal position and to select the summary statistic he is interested in.



(a) Query Builder.



(b) Result Visualizer.

Figure 5: i2b2 front-end plugin.

5.2 Performance Evaluation

To evaluate the performance of our proposed solution in the real operational setting of the Lausanne University Hospital, we have performed several tests on real cohorts of patients with clinical and genomic data.

Experimental Setup. To show the practicality of our solution, we used only commodity hardware for our experiments. The i2b2 server module and the proxy server module run on two servers in the CHUV's research network. Their configurations are described in Table 3. For both servers, we limited the number of threads to 8. At the user side, we used an off-the-shelf laptop equipped with Windows 10, intel i7-3517U processor and 10 GB of memory. We ran the i2b2 Web-client on Firefox 47.0.1.

	Data warehouse i2b2 Server	Proxy server
Operating System	Ubuntu 14.04	Ubuntu 14.04
Processor	Intel Xeon E3-1270	Intel Core i7-620M
Memory	16 GB	4 GB
Maximum Memory Usage for JVM	8 GB	512 MB
Database	PostgreSQL 9.4	MySQL 5.6

Table 3: Server Setting

We used the implementation of the FV cryptosystem provided within the NFLlib library by Aguilar-Melchor *et al.*[44]. The NFLlib is an optimized open-source C++ library dedicated to ideal lattice cryptography in the polynomial ring $\mathbb{Z}_p[x]/(x^n+1)$ for n a power of 2. We chose this library because, to the best of our knowledge, it is most efficient one for computations over polynomials. Indeed, NFLlib uses a mixed NTT-CRT representation to reduce computational costs: Number-Theoretic Transform (NTT) for polynomials[45] and Chinese Remainder Theorem (CRT) for integers.

We used the FV encryption parameters, as reported in Table 4, in order to have 128 bits of security level.

Parameter	Value
Polynomial Degree	2048
Ciphertext Modulus	4,611,686,018,326,724,609 (62 bits)
Plaintext Modulus	1,000,000 (20 bits)

Table 4: Encryption Parameters

We used real genomic data coming from the exome sequencing of 392 samples giving a genotyping for 472,845 variants each. The resulting VCF file contained a total of 185,355,240 unphased genotypes. For clinical data, we used 90,454 clinical records from 134 patients available from the i2b2 demo version[46]. Patients from the i2b2 demo were duplicated in order to match the number of patients with genomic records. As a result, we had an initial cohort of 392 individuals with both clinical and genomic data. To test the scalability of our solution, this initial cohort was further extended by replicating individuals in order to obtain a cohort of 5,000 patients.

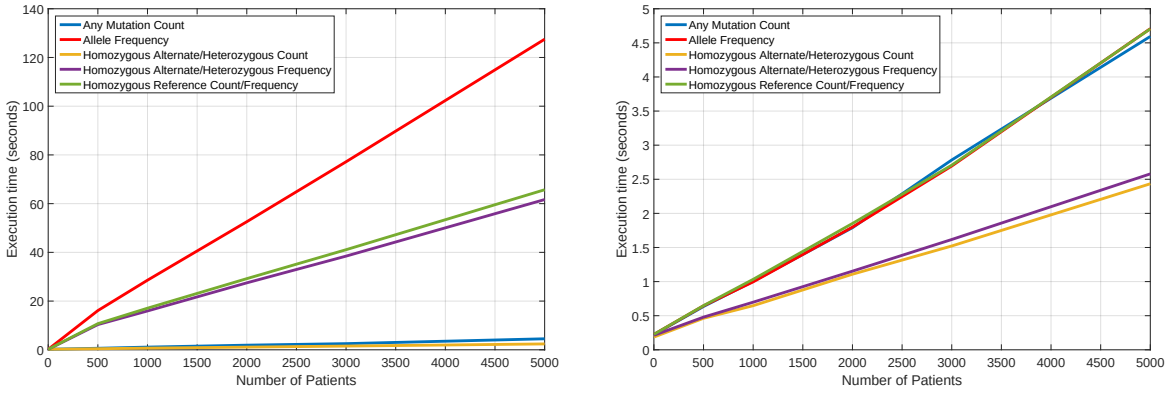
Performance Analysis. We assessed the performance of our proposed solution in terms of storage and computational overheads.

To measure the storage overhead, we compared the initial size of genotypes within the original VCF file with the size of the encrypted genotypes stored on the i2b2 server. We did not consider the meta-data in the VCF file as this information is never modified before being stored on the i2b2 server. In general, in a VCF file, a genotype is represented by 4 bytes: 2 bytes for 2 alleles, 1 byte for '/' or '|', and 1 byte for a delimiter. As there were a total of 185,355,240 genotypes in our VCF file, their corresponding size was 707.07 MB. After encoding, packing and encrypting all genotypes in the VCF file, we obtained a set of 181,010 ciphertexts whose size is 5.82 GB. This corresponds to a storage overhead of 8x compared to the unencrypted VCF.

To measure the computational overhead, we ran experiments on all the privacy-preserving algorithms for summary statistics for different sets of variants and different cohort sizes. Experiments were run 100 times for each scenario and we report the average execution time in the following. We evaluated the different steps of the query execution phase, described in detail in Section 4.4.

Fig.6a and Fig.6b show the total execution time at the i2b2 server needed to compute the different summary statistics for increasing cohort sizes and a fixed query including 3,000 genetic variants with and without *no-calls*. It is easy to observe how the execution time increases linearly with the number of patients in the cohort and how the presence of *no-calls* in the VCF file has a significant impact on performance. Differences between the execution time for computing homozygous alternate/heterozygous counts/frequencies (yellow and purple curves) and the other summary statistics are mainly due to the different number of ciphertexts involved in the computation.

The execution time at the i2b2 does not depend only on the number of patients in the cohort but also on the number of consecutive genetic variants specified in the query, as shown in Fig.7. The differences in execution time between the different summary statistics depend on genotypes with no-call values. In particular, the computations of the number of mutations (blue curve) and the number of homozygous alternate/heterozygous (yellow curve) are significantly



(a) With *no-calls*. (b) Without *no-calls*.

Figure 6: Total query execution time at i2b2 server per number of patients in the cohort for a query involving 3000 genetic variants.

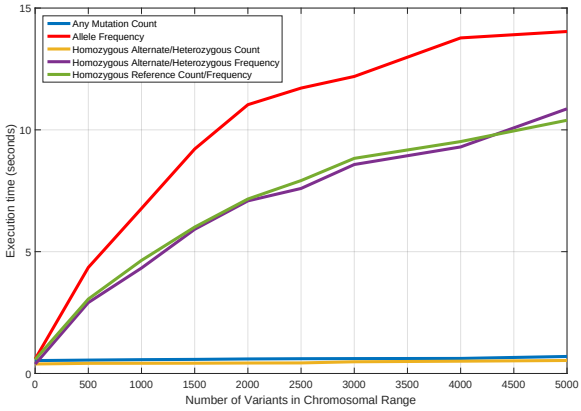


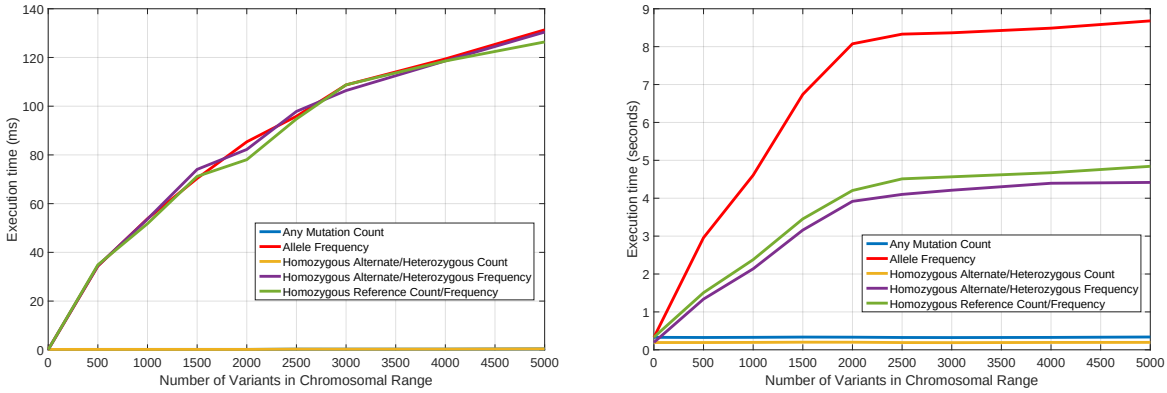
Figure 7: Total execution time at i2b2 server per number of consecutive genetic variants.

influenced only by the existence of genotypes with 1 no-call, whereas the other computations are also influenced by genotypes with 2 no-calls. Note that in our data we do not have genotypes with 1 no-call. From Fig.8a and Fig.8b we can observe that most of the computational time at the i2b2 server is due to data retrieval from the database and homomorphic computations.

Finally, Fig.9a and Fig.9b show the execution time for a full decrpytion of the query results for an increasing number of consecutive genetic variants for the contribution of the proxy server and the client, respectively. It is easy to observe that the execution time at the proxy server is similar to the execution time at the i2b2 server for partial decryption as the number of processed ciphertexts is the same. At the client side, we can observe a different behavior for the execution time as decryption is done variant-by-variant instead of ciphertext-by-ciphertext. Note that, because of genotypes with no-call values, the number of ciphertexts including the query results can be different from the number of ciphertexts including genetic variants.

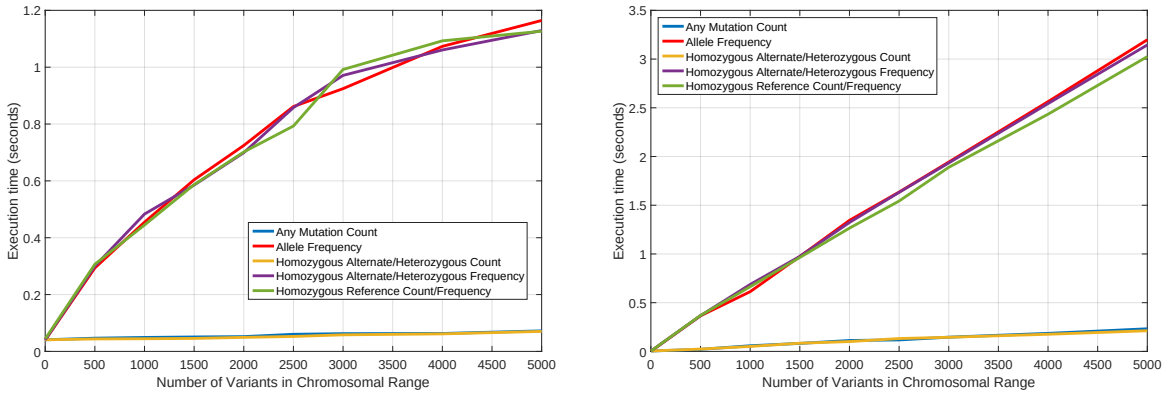
6 Discussion

In this paper, we have described how we designed, implemented and deployed, for the first time, a secure and privacy-preserving solution for exploring genomic cohorts in a real operational scenario at the Lausanne University Hospital. We have thoroughly evaluated the performance of our solution on real data. Results show that in general privacy-preserving solutions, such as the one proposed in this work, can already be used in medical settings as new efficient



(a) Partial decryption. (b) Data retrieval and homomorphic computations.

Figure 8: Breakdown of total execution time at i2b2 server per number of consecutive genetic variants.



(a) At proxy server. (b) At client.

Figure 9: Execution time for full decryption for queries with increasing number of consecutive genetic variants.

enablers. Yet, some important points need to be further discussed.

As it can be easily observed from the results of Fig. 6a and Fig. 6b, the main bottleneck for the execution time of queries involving specific types of summary statistics (e.g., allele or genotype frequencies) is due to the presence of genotypes with *no-call* values. Indeed, because of our encoding that minimizes the storage cost (as required by CHUV that plans to encrypt thousands of variants for thousands of patients), *no-calls* have a significant impact on performance. Yet, some alternative approaches can be used to easily address this issue. A first potential approach consists in using a different encoding for genotype values, as the one shown in Table 5, which maximizes performance at the expense of increasing the storage overhead from a factor of 8 to a factor of 20. Another potential approach would be to perform genotype imputation before the genotype encoding in order to replace no-calls with imputed values at the expense of a slight decrease in accuracy. This said, it is easy to observe how the first alternative prioritizes high performance and high accuracy instead of low storage overhead, whereas the second approach ensures high performance and low storage overhead rather than full accuracy. Note that, by slightly sacrificing performance, our current solution assigns the highest priority to low storage cost and high accuracy.

Genotype value	Genotype encoding				
	gt1	*gt2*	*gt3*	*gt4*	*gt5*
./.	0	0	0	0	0
./0	1	0	0	0	0
./1	0	1	0	0	0
0/0	0	0	1	0	0
0/1	0	0	0	1	0
1/1	0	0	0	0	1

Table 5: Genotype encoding optimizing performance.

As it is well-known in the security field, the perfect solution does not exist. It is always a matter of finding the best trad-offs between protection cost, efficiency, and accuracy of the result. Our proposed solution is general enough to be fine-tuned according to the requirements.

As initially mentioned in Section 1, the proposed solution addresses a simple use case by providing genomic summary statistics that can be securely computed only through homomorphic additions. Yet, thanks to the flexibility of the FV scheme, more complex use cases such as privacy-preserving phenome-wide association studies (PheWAS) or genome-wide association studies (GWAS) could also be developed on top of our solution. We are planning to explore the feasibility of more complex use cases in future work. Finally, we want to emphasize that the goal of this paper was not to discuss the parameters' values that determine the best privacy and accuracy trade-off when using differential privacy but to provide the tools that enable privacy-preserving exploration of genetic cohorts. We believe that such a discussion should be a prerogative of database administrators, end-users and hospitals' IRBs rather than a part of our solution.

Acknowledgments

The authors would like to thank Sebastien Rocher and Gianni Benezet, from CHUV's data-warehouse team, for their precious and valuable feedback and their support during the deployment phase at CHUV of the proposed solution, Patrick Zosso from the Informatics Department of CHUV, for his help in setting up the i2b2 and proxy server and, finally, Carlos Aguilar-Melchor from the University of Toulouse, France, and Marc-Olivier Killijian from CNRS, Toulouse, France, for their instrumental help with the NFLlib library.

References

1. Service de soutien à la recherche clinique chuv. `http://www.chuv.ch/chuv_home/recherche/chuv_recherche-infos-pratiques/chuv_recherche-infos-pratiques-src.htm`.

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

3. Naveed Muhammad and *et al.* . Privacy in the Genomic Era. *Accepted to ACM Computing Surveys*, 2015.

4. Murphy Shawn N, Weber Griffin, Mendis Michael, Gainer Vivian, Chueh Henry C, Churchill Susanne et al. Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). *Journal of the American Medical Informatics Association*, 17(2):124–130, 2010.

5. i2b2 installations, . URL `https://www.i2b2.org/work/i2b2_installations.html`.

6. Homer Nils, Szelinger Szabolcs, Redman Margot, Duggan David, Tembe Waibhav, Muehling Jill et al. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS Genetics*, 4(8), 2008.

7. Shringarpure Suyash S and Bustamante Carlos D. Privacy risks from genomic data-sharing beacons. *The American Journal of Human Genetics*, 97(5):631–646, 2015.

8. Gymrek Melissa, McGuire Amy L, Golan David, Halperin Eran and Erlich Yaniv. Identifying personal genomes by surname inference. *Science*, 339(6117):321–324, 2013.

9. Humbert Mathias, Huguenin Kévin, Hugonot Joachim, Ayday Erman and Hubaux Jean-Pierre. De-anonymizing genomic databases using phenotypic traits. In *15th Privacy Enhancing Technologies Symposium (PETS)*, 2015.

10. Malin Bradley and Sweeney Latanya. 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*, 37(3):179–192, 2004.

11. Wang Rui, Li Yong Fuga, Wang XiaoFeng, Tang Haixu and Zhou Xiaoyong. Learning your identity and disease from research papers: Information leaks in genome wide association study. In *CCS*, pages 534–544, 2009.

12. Zhou Xiaoyong, Peng Bo, Li Yong Fuga, Chen Yangyi, Tang Haixu and Wang XiaoFeng. To release or not to release: Evaluating information leaks in aggregate human-genome data. In *ESORICS*, 2011.

13. Canim M., Kantarcioglu M. and Malin B. Secure management of biomedical data with cryptographic hardware. *Information Technology in Biomedicine, IEEE Transactions on*, 16(1):166–175, 2012. ISSN 1089-7771.

14. Kamm Liina, Bogdanov Dan, Laur Sven and Vilo Jaak. A new way to protect privacy in large-scale genome-wide association studies. *Bioinformatics*, 2013.

15. Xie Wei, Kantarcioglu Murat, Bush William S., Crawford Dana, Denny Joshua C., Heatherly Raymond et al. SecureMA: protecting participant privacy in genetic association meta-analysis. *Bioinformatics (Oxford, England)*, 2014.

16. Wang Xiao Shaun, Huang Yan, Zhao Yongan, Tang Haixu, Wang XiaoFeng and Bu Diyue. Efficient genome-wide, privacy-preserving similar patient query based on private edit distance. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 492–503. ACM, 2015.

17. Kantarcioglu M., Jiang Wei, Liu Ying and Malin B. A cryptographic approach to securely share and query genomic sequences. *Information Technology in Biomedicine, IEEE Transactions on*, 12(5):606–617, 2008. ISSN 1089-7771. doi: 10.1109/TITB.2007.908465.

18. Lu Wenjie, Yamada Yoshiji and Sakuma Jun. Efficient secure outsourcing of genome-wide association studies. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 3–6. IEEE, 2015.

19. Duverle David A, Kawasaki Shohei, Yamada Yoshiji, Sakuma Jun and Tsuda Koji. Privacy-preserving statistical analysis by exact logistic regression. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 7–16. IEEE, 2015.

20. Wang Shuang, Jiang Xiaoqian, Fox Dov and Ohno-Machado Lucila. Preserving genome privacy in research studies. In *Medical Data Privacy Handbook*, pages 425–441. Springer, 2015.

21. Wang Shuang, Zhang Yuchen, Dai Wenrui, Lauter Kristin, Kim Miran, Tang Yuzhe et al. Healer: Homomorphic computation of exact logistic regression for secure rare disease variants analysis in gwas. *Bioinformatics*, 32(2):211–218, 2016.

22. Uhlerop Caroline, Slavković Aleksandra and Fienberg Stephen E. Privacy-preserving data sharing for genome-wide association studies. *The Journal of privacy and confidentiality*, 5(1):137, 2013.

23. Johnson Aaron and Shmatikov Vitaly. Privacy-preserving data exploration in genome-wide association studies. In *KDD*, pages 1079–1087, 2013.

24. Yu Fei, Fienberg Stephen E., Slavkovic Aleksandra B. and Uhler Caroline. Scalable privacy-preserving data sharing methodology for genome-wide association studies. *Journal of Biomedical Informatics*, 2014.

25. Simmons Sean and Berger Bonnie. One size doesn't fit all: Measuring individual privacy in aggregate genomic data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 41–49. IEEE, 2015.

26. Simmons Sean and Berger Bonnie. Realizing privacy preserving genome-wide association studies. *Bioinformatics*, 32(9):1293–1300, 2016.

27. Simmons Sean, Sahinalp Cenk and Berger Bonnie. Enabling privacy-preserving gwass in heterogeneous human populations. *Cell Systems*, 3 (1):54 – 61, 2016.

28. Tramèr Florian, Huang Zhicong, Hubaux Jean-Pierre and Ayday Erman. Differential privacy with bounded priors: Reconciling utility and privacy in genome-wide association studies. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1286–1297. ACM, 2015.

29. McLaren Paul J., Raisaro Jean Louis, Aouri Manel, Rotger Margalida, Ayday Erman, Bartha István et al. Privacy-preserving genomic testing in the clinic: a model using HIV treatment. *Genetics in Medicine*, 18(8):814–822, aug 2016. ISSN 1098-3600. doi: 10.1038/gim.2015.167. URL `http://www.nature.com/doifinder/10.1038/gim.2015.167`.

30. Phillips Lori C, Minovitsky Simon, Ratnere Igor, Dubchak Inna, Kohane I and Murphy SN. Use genomic variants in informatics for integrating biology and the bedside (i2b2). *AMIA Proceedings Library: 48-52 T2011*, 2011.

31. Gabetta Matteo, Limongelli Ivan, Rizzo Ettore, Riva Alberto, Segagni Daniele and Bellazzi Riccardo. Bigq: a nosql based framework to handle genomic variants in i2b2. *BMC bioinformatics*, 16(1):1, 2015.

32. Murphy Shawn N, Gainer Vivian, Mendis Michael, Churchill Susanne and Kohane Isaac. Strategies for maintaining patient privacy in i2b2. *Journal of the American Medical Informatics Association*, 18(Supplement 1):i103–i108, 2011.

33. Gentry Craig. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

34. Fan Junfeng and Vercauteren Frederik. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012. URL `http://eprint.iacr.org/2012/144`.

35. Koblitz Neal. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

36. Bos Joppe W., Lauter Kristin E., Loftus Jake and Naehrig Michael. Improved security for a ring-based fully homomorphic encryption scheme. In *Cryptography and Coding - 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings*, pages 45–64, 2013.

37. Brakerski Zvika, Gentry Craig and Halevi Shai. Packed ciphertexts in lwe-based homomorphic encryption. In *Public-Key Cryptography–PKC 2013*, pages 1–13. Springer, 2013.

38. Dwork Cynthia. Differential privacy. Venice, Italy, July 2006. Springer Verlag. URL `https://www.microsoft.com/en-us/research/publication/differential-privacy/`.

39. The variant call format specification - vcfv4.3 and bcfv2.2. `http://samtools.github.io/hts-specs/VCFv4.3.pdf`. Accessed: 2016-08-02.

40. Schlesinger Jennifer. Dark web is fertile ground for stolen medical records. URL `http://www.cnbc.com/2016/03/10/dark-web-is-fertile-ground-for-stolen-medical-records.html`.

41. Global Alliance for Genomics and Health . The beacon project. URL `https://beacon-network.org/#/`.

42. Exac browser. URL `http://exac.broadinstitute.org`.

43. Namazi Mina, Troncoso-Pastoriza Juan Ramón and Pérez-González Fernando. Dynamic privacy-preserving genomic susceptibility testing. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, pages 45–50. ACM, 2016.

44. Aguilar-Melchor Carlos, Barrier Joris, Guelton Serge, Guinet Adrien, Killijian Marc-Olivier and Lepoint Tancrede. Nfllib: Ntt-based fast lattice library. In *Cryptographers' Track at the RSA Conference*, pages 341–356. Springer, 2016.

45. Göttert Norman, Feller Thomas, Schneider Michael, Buchmann Johannes and Huss Sorin. On the design of hardware building blocks for modern lattice-based encryption schemes. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 512–529. Springer, 2012.

46. i2b2 software, . URL `https://www.i2b2.org/software/index.html`.

47. Melchor Carlos Aguilar, Barrier Joris, Guelton Serge, Guinet Adrien, Killijian Marc-Olivier and Lepoint Tancrède. Nfllib: Ntt-based fast lattice library. In Sako Kazue, editor, *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, volume 9610 of *Lecture Notes in Computer Science*, pages 341–356. Springer, 2016. ISBN 978-3-319-29484-1. doi: 10.1007/978-3-319-29485-8_20. URL `http://dx.doi.org/10.1007/978-3-319-29485-8_20`.

48. Dowlin Nathan, Gilad-Bachrach Ran, Laine Kim, Lauter Kristin, Naehrig Michael and Wernsing John. Manual for using homomorphic encryption for bioinformatics. Technical Report MSR-TR-2015-87, November 2015. URL `http://research.microsoft.com/apps/pubs/default.aspx?id=258435`.

7 Appendix

7.1 Homomoprhic Encryption Scheme

In order to choose the best cryptosystem for our solution in terms of storage and computational overheads, we compared three C++ libraries: Elliptic curve ElGamal (EC-ElGamal)[35], Fan and Vercauteren (FV) scheme[34] with NTT-based fast lattice library (NFLlib)[47], and Simple Encrypted Arithmetic Library (SEAL)[48] which is an implementation of Yet Another Somewhat Homomorphic Encryption (YASHE) scheme[36]. We compared the performance of these libraries on a laptop with the i7-3517U processor and 10 GB of RAM. The results are as on Table 6 and Table 7. Based on this benchmark, we chose the FV scheme with NFLlib for our solution.

	EC-ElGamal (Java)	FV + NFLlib (C++)		SEAL (C++)	
		No Packing	Packing (n=4096)	No Packing	Packing (n=4096)
Ciphertext Size	360 Bytes	128 KB	128 KB	64 KB	64 KB
Number of Required Ciphertext	1 million	1 million	245	1 million	245
Storage Size	343.3 MB	122 GB	30.6 MB	61 GB	15.5 MB

Table 6: Storage requirement to encrypt 1 million variants with 256 bits of security level

	EC-ElGamal	FV + NFLlib	SEAL
Encryption	4 ms	1 ms	71 ms
Decryption	10 ms	2.5 ms	70 ms
Homomorphic Addition	0.2 ms	0.1 ms	0.2 ms
Homomorphic Multiplication	-	18 ms	1286 ms

Table 7: Execution time of each operation

7.2 Secure Algorithms for Computation of Genomic Summary Statistics

In the followings, we describe how summary statistics outlined in Section 4.3 can be securely computed at the i2b2 server by using homomorphic encryption. Differential privacy can then be satisfied by homomorphically adding noise to the encrypted result.

Algorithm 2: Count any mutation

Input : Patient set P, variant set V which are in the same ciphertext, set of the number of no calls N_p of each patient $p \in P$, and set of encrypted genotypes G_p of each patient $p \in P$. G_p consists of two ciphertexts as $G_p.cipher\_gt1$ and $G_p.cipher\_gt2$

Output: Map between the set of variants and the result ciphertext

$M \leftarrow$ an empty map // Map of key/value pairs containing encrypted results per set of variants which can be computed together

$M.addEntry(V, \mathsf{Enc}(0))$

for $p \in P$ **do**

  ones $\leftarrow$ Set of variants in V whose $N_p = 1$

  $M' \leftarrow$ an empty map

  for $(key, value) \in M$ **do**

    if $key \cap ones \neq \varnothing$ **then**

      $M'.addEntry(key \cap \text{ones}, value \oplus G_p.cipher\_gt1)$

        // Computation for genotype ./0 and ./1

    end

    if $key \setminus ones \neq \varnothing$ **then**

      $M'.addEntry(key \setminus \text{ones}, value \oplus G_p.cipher\_gt1 \oplus G_p.cipher\_gt2)$

        // Computation for genotype ./., 0/0, 0/1 and 1/1

    end

  end

  $M \leftarrow M'$

end

return M

Algorithm 3: Homozygous alternate or heterozygous count

Input : Patient set P, variant set V which are in the same ciphertext, set of the number of no calls N_p of each patient $p \in P$, and set of encrypted genotypes G_p of each patient $p \in P$. G_p consists of two ciphertexts as $G_p.cipher\_gt1$ and $G_p.cipher\_gt2$

Output: Map between the set of variants and the result ciphertext

$M \leftarrow$ an empty map // Map of key/value pairs containing encrypted results per set of variants which can be computed together

$M.addEntry(V, \mathsf{Enc}(0))$

for $p \in P$ **do**

  ones $\leftarrow$ Set of variants in V whose $N_p = 1$

  $M' \leftarrow$ an empty map

  for $(key, value) \in M$ **do**

    if $key \cap ones \neq \varnothing$ **then**

      $M'.addEntry(key \cap \text{ones}, value)$ // Computation for genotype ./0 and ./1

    end

    if $key \setminus ones \neq \varnothing$ **then**

      $M'.addEntry(key \setminus \text{ones}, value \oplus G_p.cipher\_gt2)$ ▷ $cipher\_gt1$ instead of $cipher\_gt2$ for heterozygous

        // Computation for genotype ./., 0/0, 0/1 and 1/1

    end

  end

  $M \leftarrow M'$

end

return M

Algorithm 4: Homozygous alternate or heterozygous frequency

Input : Patient set P, variant set V which are in the same ciphertext, set of the number of no calls N_p of each patient $p \in P$, and set of encrypted genotypes G_p of each patient $p \in P$. G_p consists of two ciphertexts as $G_p.cipher\_gt1$ and $G_p.cipher\_gt2$

Output: Map between the set of variants and the result ciphertext

$M \leftarrow$ an empty map // Map of key/value pairs containing encrypted results per set of variants which can be computed together

$M.addEntry(V, (\mathsf{Enc}(0), 0))$

for $p \in P$ **do**

  zeros $\leftarrow$ Set of variants in V whose $N_p = 0$

  $M' \leftarrow$ an empty map

  for $(key, value) \in M$ **do**

    if $key \setminus zeros \neq \varnothing$ **then**

      $M'.addEntry(key \setminus \text{zeros}, value)$ // Computation for genotype ./., ./0 and ./1

    end

    if $key \cap zeros \neq \varnothing$ **then**

      $M'.addEntry(key \cap \text{zeros}, (value[0] \oplus G_p.cipher\_gt2, value[1] + 1)$ // Computation for genotype 0/0, 0/1 and 1/1

      ▷ $cipher\_gt1$ instead of $cipher\_gt2$ for heterozygous

    end

  end

  $M \leftarrow M'$

end

return M ▷ $value[0]/value[1]$ will be computed at the client side

Algorithm 5: Homozygous reference count or frequency

Input : Patient set P, variant set V which are in the same ciphertext, set of the number of no calls N_p of each patient $p \in P$, and set of encrypted genotypes G_p of each patient $p \in P$. G_p consists of two ciphertexts as $G_p.cipher\_gt1$ and $G_p.cipher\_gt2$

Output: Map between the set of variants and the result ciphertext

$M \leftarrow$ an empty map // Map of key/value pairs containing encrypted results per set of variants which can be computed together

$M.addEntry(V, (\mathsf{Enc}(0), 0))$

for $p \in P$ **do**

  zeros $\leftarrow$ Set of variants in V whose $N_p = 0$

  $M' \leftarrow$ an empty map

  for $(key, value) \in M$ **do**

    if $key \setminus zeros \neq \varnothing$ **then**

      $M'.addEntry(key \setminus \text{zeros}, value)$ // Computation for genotype ./., ./0 and ./1

    end

    if $key \cap zeros \neq \varnothing$ **then**

      $M'.addEntry(key \cap \text{zeros}, (value[0] \oplus G_p.cipher\_gt1 \oplus G_p.cipher\_gt2, value[1] + 1))$ // Computation for genotype 0/0, 0/1 and 1/1

    end

  end

  $M \leftarrow M'$

end

for $(key, value) \in M$ **do**

  $value[0] \leftarrow \mathsf{Enc}(value[1] + value[1]x + value[1]x^2 + \cdots + value[1]x^{\ell-1}) \oplus -value[0]$

end

return M ▷ $value[0]/value[1]$ will be computed at the client side for the homozygous reference frequency