Privacy-Preserving Processing of Raw Genomic Data

Erman Ayday[1], Jean Louis Raisaro[1], Urs Hengartner[2], Adam Molyneaux[3], and Jean-Pierre Hubaux[1]

[1] École Polytechnique Fédérale de Lausanne
[2] University of Waterloo
[3] Sophia Genetics

Abstract. Geneticists prefer to store patients' aligned, raw genomic data, in addition to their variant calls (compact and summarized form of the raw data), mainly because of the immaturity of bioinformatic algorithms and sequencing platforms. Thus, we propose a privacy-preserving system to protect the privacy of aligned, raw genomic data. The raw genomic data of a patient includes millions of short reads, each comprised of between 100 and 400 nucleotides (genomic letters). We propose storing these short reads at a biobank in encrypted form. The proposed scheme enables a medical unit (e.g., a pharmaceutical company or a hospital) to privately retrieve a subset of the short reads of the patients (which include a definite range of nucleotides depending on the type of the genetic test) without revealing the nature of the genetic test to the biobank. Furthermore, the proposed scheme lets the biobank mask particular parts of the retrieved short reads if (i) some parts of the provided short reads are out of the requested range, or (ii) the patient does not give consent to some parts of the provided short reads (e.g., parts revealing sensitive diseases). We evaluate the proposed scheme to show the amount of unauthorized genomic data leakage it prevents. Finally, we implement the proposed scheme and assess its practicality.

Keywords: Genomics, Privacy, Bioinformatics, Raw genomic data

1 Introduction

Genomics holds great promise for better predictive medicine and improved diagnoses. However, genomics also comes with a risk to privacy [1, 2] (e.g., revelation of an individual's genetic properties due to the leakage of his genomic data). An increasing number of medical units (pharmaceutical companies or hospitals) are willing to outsource the storage of genomes generated in clinical trials. Acting as a third party, a biobank could store patients' genomic data that would be used by the medical units for clinical trials. In the meantime, the patient can also benefit from the stored genomic information by interrogating his own genomic data, together with his family doctor, for specific genetic predispositions, susceptibilities and metabolical capacities. The major challenge here is to preserve the privacy of patients' genomic data while allowing the medical units to operate on specific parts of the genome (for which they are authorized).

Sequence alignment/map (SAM and its binary version BAM) files are the *de facto* standards used to store the aligned[4], raw genomic data generated by next-generation DNA sequencers and bioinformatic algorithms. There are hundreds of millions of short reads (each including between 100 and 400 nucleotides) in the SAM file of a patient. Typically, each nucleotide is present in several short reads in order to have sufficiently high coverage of each patient's DNA. In the rest of this paper, we present our work focusing on the SAM files, as it is clearer to present the proposed methods by using this human-readable format. However, the proposed

[4] Alignment is with respect to the reference genome, which is assembled by the scientists.

scheme has no reliance on this particular format; our proposed algorithms can also be applied to other data formats that are used to store the raw genomic data (e.g., BAM).

In general, geneticists prefer storing aligned, raw genomic data of the patients (i.e., their SAM files), in addition to their variant calls (which include each nucleotide on the DNA sequence once, hence is much more compact) due to the following reasons: (i) Bioinformatic algorithms and sequencing platforms for variant calling are currently not yet mature, and hence geneticists prefer to observe each nucleotide in several short reads. (ii) If a patient carries a disease, which causes specific variations in the diseased cells (e.g., cancer), his DNA sequence in his healthy cells will be different from those diseased. Such variations can be misclassified as sequencing errors by only looking at the patient's variant calls (rather than his short reads). And (iii) due to the rapid evolution of genomic research, geneticists do not know enough to decide which information should really be kept and what is superfluous, hence they prefer to store all outcome of the sequencing process as SAM files.

To the best of our knowledge, none of the existing works on genomic privacy addresses the issue of private processing of aligned, raw genomic data (i.e., SAM files), which is crucial to enable the use of genomic data in clinical trials. Therefore, in this paper, we propose a privacy-preserving system for the storage, retrieval and processing of the SAM files. In a nutshell, the proposed scheme stores the encrypted SAM files of the patients at a *biobank* and it provides the requested range of nucleotides (on the DNA sequence) to a medical unit while protecting the patients' genomic privacy. It is important to note that the proposed scheme enables the privacy-preserving processing of the SAM files both for individual treatment (when the medical unit is embodied in a hospital) and for genetic research (when the medical unit is embodied in a pharmaceutical company). The main contributions of this paper are summarized in the following:

1. We develop a privacy-preserving framework for the retrieval of encrypted short reads (in the SAM files) from the biobank without revealing the scope of the request to the biobank.
2. We develop an efficient system for obfuscating (i.e., masking) specific parts of the encrypted short reads that are out of the requested range of the medical unit (or that the patient prefers to keep secret) at the biobank before providing them to the medical unit.
3. We show the benefit of masking by evaluating the information leak to the medical unit, with and without the masking is in place.
4. We implement the proposed privacy-preserving system by using real genomic data, evaluate its efficiency, and show its practicality.

The rest of the paper is organized as follows: In the next section, we summarize the existing work on genomic privacy. In Section 3, we give a brief background on genomics (particularly on SAM files). In Section 4, we give an overview of the proposed scheme. In Section 5, we discuss the potential options and constraints about the design of our proposed scheme. In Section 6, we discuss the threat model and our security considerations. In Section 7, we describe the proposed scheme in detail. In Section 8, we evaluate our proposed scheme using real genomic data. In Section 9, we discuss about the implementation of the proposed scheme and its practicality. In Section 10, we conclude the paper.

2 Related Work

We can put the research on genomic privacy in three main categories: (i) re-identification of anonymized genomic data, (ii) cryptographic algorithms to protect genomic data, and (iii) private clinical genomics.

Homer *et al.* [3] prove that de-identification is an ineffective way to protect the privacy of genomic data, which is also supported by other works [4–6]. Most recently, Gymrek *et al.* [7] show how they identified DNAs of several individuals who participated in scientific studies. Fienberg *et al.* [8] propose using differential privacy to protect the identities of scientific study participants, however this approach reduces the accuracy of the research results. Some pieces of work also focus on protecting the privacy of genomic data and on preserving utility in medical tests such as (i) search of a particular pattern in the DNA sequence [9,10], (ii) comparing the similarity of DNA sequences [11–13], and (iii) performing statistical analysis on several DNA sequences [14]. Moreover, we propose privacy-preserving schemes for medical tests and personalized medicine methods that use patients' genomic data [15–17].[5] For privacy-preserving clinical genomics, a group of researchers proposes to outsource some costly computations to a public cloud or semi-trusted service provider [18, 19].

As we discussed before, none of the aforementioned efforts on genomic privacy focus on the processing of aligned, raw genomic data. Therefore, in this work, we focus on private storage, retrieval, and processing of raw genomic data.

3 Genomic Background

The DNA sequence data produced by DNA sequencing consists of millions of short reads, each typically including between 100 and 400 nucleotides (A,C,G,T), depending on the type of sequencer. These reads are randomly sampled from a human genome. Each read is then bioinformatically treated and positioned (aligned) to its genetic location to produce a so-called SAM file. There are hundreds of millions of short reads in the SAM file of one patient.

The privacy-sensitive fields of a short read are (i) its position with respect to the reference genome, (ii) its *cigar string* (CS), and (iii) its content (including the nucleotides from $\{A, T, G, C\}$).

A short read's position denotes the position of the first aligned nucleotide in its content, with respect to the reference genome. The position of a short read is in the form $L_{i,j} = \langle x_i | y_j \rangle$, where x_i represents the chromosome number ($x_i \in [1, 23]$ as there are 23 chromosomes in the human genome) and y_j represents the position of its first aligned nucleotide on chromosome x_i ($y_j \in [1, 240\text{M}]$ as the maximum number of nucleotides on a chromosome is around 240 million). The cigar string (CS) of a short read expresses the variations in the content of the short read. The CS includes *pairs* of nucleotide lengths and the associated operations. The operations in the CS indicate some properties about content of the short read such as which nucleotides align with the reference, which are deleted from the reference, and which are insertions that are not in the reference (without revealing the content of the short read). Finally, the content of a short read includes the nucleotides. We provide more details about the SAM files in [20]. That is, we illustrate the real format of a short read, we give the descriptions of the fields in the CS and we give

[5] More information about our activities in the field of genomic privacy can be found at: http://lca.epfl.ch/projects/genomic-privacy/.

an example about how the content of a short read looks and how the CS of the corresponding short read is generated.

There are several types of DNA variations in the human genome, among which the *single nucleotide polymorphism* (SNP) is the most common. A SNP is a position in the genome holding a nucleotide that varies between individuals. Recent discoveries show that the susceptibility of a patient to several diseases can be computed from his SNPs [21]. Thus, we focus on the SNPs of a patient when evaluating the information leakage in Section 8.

4 Overview of the Proposed Solution

In this work, we develop a privacy-preserving system for the storage, retrieval and processing of the SAM files (details are in Section 7).

We assume that the sequencing and encryption of the genomes are done at a *certified institution* (CI), which is a trusted entity. We note that having such a trusted entity cannot be avoided as the sequencing has to be done at some institution to obtain the SAM files of the patients. Each part (position, CS, and content) of each short read (in the SAM file) is encrypted (via a different encryption scheme) after the sequencing, and encrypted SAM files of the patients are stored at a biobank. We assume that SAM files are stored at the biobank by using pseudonyms; this way, the biobank cannot associate the conducted genetic tests and the *medical unit* (MU), which conduct these tests, with the real identities of the patients. We note that a private company (e.g., cloud storage service) or the government could play the role of the biobank. There are potentially multiple MUs in the system, and each MU is an approved institution (by the medical authorities). Furthermore, we assume that an MU is a broad unit consisting of many sub-units (e.g., physicians or specialized clinics) that can potentially request nucleotides from any parts of a patient's genome.

The cryptographic keys of the patients are stored on a key manager by using the patient's pseudonym (which does not require the participation of the patient in the protocol). From here on, we assume the existence of a *masking and key manager* (MK) in the system to store the cryptographic keys of the patients. The MK can also be embodied in the government or a private company.

4.1 Privacy-Preserving Retrieval of the Short Reads

When the MU requests a specific range of nucleotides (on the DNA sequence of one or multiple patients), the biobank provides all the short reads that include at least one nucleotide from the requested range through the MK. During this process, the patient does not want to reveal his complete genome to the MU, to the biobank, or to the MK. Furthermore, it is not desirable for the biobank to learn the requested range of nucleotides (as the biobank can infer the nature of the genetic test from this requested range). Thus, we develop a privacy-preserving system for the retrieval of the short reads by the MU. The proposed scheme provides the short reads that include the requested range of nucleotides to the MU without revealing the positions of these short reads to the biobank.

To achieve this goal, we first modify the structure of the genome by permuting the positions of the short reads, and then we use order preserving encryption (OPE) on the positions of the short reads (in the SAM file). OPE is a deterministic encryption scheme whose encryption function preserves numerical ordering of the

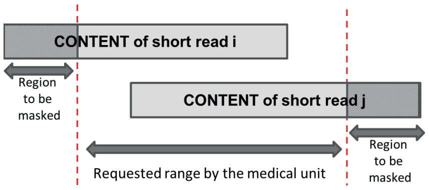


Fig. 1. Parts to be masked in the short reads for out-of-range content.

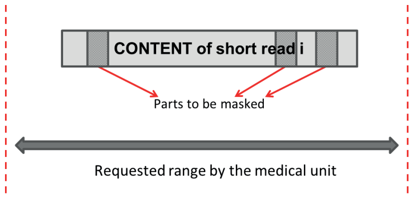


Fig. 2. Parts to be masked in a short read based on patient's consent. The patient does not give consent to reveal the dark parts of the short read.

plaintexts [22,23].[6] Thus, OPE enables the encryption of the positions of the short reads and preserves the numerical ordering of the plaintext positions.

4.2 Masking of the Short Reads

We prevent the leakage of extra information in the short reads to the MU by masking the encrypted short reads at the biobank (before sending them to the MU). As each short read includes between 100 and 400 nucleotides, some provided short reads might include information out of the MU's requested range of genomic data, as in Fig. 1. Similarly, some provided short reads might contain privacy-sensitive SNPs of the patient (which would reveal the patient's susceptibilities to privacy-sensitive diseases such as Alzheimer's), hence the patient might not give consent to reveal such parts, as in Fig. 2. From here on, the nucleotides that the patient does not consent to reveal will be referred to as the *non-consented* nucleotides.

To achieve this goal, we mask certain parts of the encrypted short reads at the biobank, without decrypting them using an efficient algorithm. It is important to note that after the short reads are decrypted at the MU, the MU is not able to determine the nucleotides at the masked positions.

5 Design Constraints and Options

For security, efficiency, and availability, we propose storing the SAM files at a biobank instead of at the MU. Extreme precaution is needed for the storage of genomic data due to its sensitivity. We assume that the biobank is more "security-aware" than an MU, hence it can protect the stored genomic data against a hacker better than an MU (yet, attacks against the biobank cannot be ruled out, as we discuss next). Indeed, this assumption is supported by recent serious medical data breaches from various MUs [24]. Furthermore, by storing the SAM files at one biobank, multiple MUs can reliably access the patients' genomic data from it (instead of each MU individually storing that same large amount of data) at any time.

We propose outsourcing the storage of the cryptographic keys (of the patients) to the MK instead of storing them on a *patient's device* (e.g., a smartphone) due to the following two reasons: (i) It is not realistic to assume that all the patients will have the sufficient precautions to protect their cryptographic keys (which will possibly be stored in their smartphones), and (ii) if the keys are stored on a patient's device, operations involving the patient are done on the MU's (e.g., the hospital) computer via the patient's device, hence this approach requires the involvement

[6] We briefly present the cryptographic tools we use in this paper in [20].

of the patient in the operation (e.g., physical presence at the hospital). Whereas, following our discussions with geneticists and medical doctors, we conclude that the patient's involvement in the genetic tests is not desired for the practicality of the protocol (e.g., when a pharmaceutical company conducts genetic research on thousands of patients).

In this work, we use OPE instead of private information retrieval (PIR), searchable encryption [25, 26], or oblivious RAM (O-RAM) storage [27] techniques for the privacy-preserving retrieval of the short reads for the following reasons: (i) As we discussed before, the short reads are randomly sampled from the genomes of the patients, and hence the positions of the short reads vary in each patient's genome. The MU typically asks for a particular range of nucleotides on the DNA sequence of one or multiple patients. However, these requested nucleotides reside in different short reads for each patient and the MU does not know which nucleotide is stored in which short reads of each patient (storing the positions of all short reads and the list of nucleotides they accommodate for each patient at every MU requires significant storage overhead). Thus, the MU does not know exactly which short reads to ask for, and hence PIR or searchable encryption techniques would be impractical for our scenario. And (ii) although O-RAM techniques completely hide the data access patterns from the server (biobank), even the most efficient implementations of O-RAM introduce high storage overhead to the client (patient) and introduce about 25 times more overhead with respect to non-oblivious storage [28].

6 Threat Model and Security Considerations

We consider the following models for the attacker:

- A curious party at the biobank (or a hacker who breaks into the biobank), who tries (i) to infer the genomic sequence of a patient from his stored genomic data and (ii) to associate the type of the genetic test (e.g., the disease for which the patient is being tested, which can be inferred from the nucleotides requested by the MU) with the patient being tested.
- A curious party at the MK (or a hacker who breaks into the MK), who tries (i) to infer the genomic sequence of a patient from his stored cryptographic keys and the information provided by the biobank and (ii) to associate the type of the genetic test with the patient being tested.
- A curious party at an MU, who can be considered either as an attacker who hacks into the MU's system or a disgruntled employee who has access to the MU's database. The goal of such an attacker is to obtain the private genomic data of a patient for which it is not authorized.

Apart from (potentially) being curious, we assume that the biobank, the MK, and the MUs are honest organizations. That is, the biobank, the MK, and the MUs honestly follow the protocols and provide correct information to the other parties. In the following, we discuss how we prevent the aforementioned attacks.

SAM files are encrypted (at the CI) and stored at the biobank to avoid the biobank from inferring the genomic data of the patients (details about encryption are in Section 7.1). To avoid the biobank from associating the conducted genetic tests with the patients, we hide both the real identities of the patients (using pseudonyms) and the types of the conducted tests (using OPE on the positions of the short reads) from the biobank. Note, however, that the biobank knows the real identity of an MU to make sure that the request comes from a valid source.[7] To avoid the MK from associating the genetic tests with the patients, we do not reveal the identities of the MUs or the patients to the MK. Alternatively (to further increase the security of the scheme), a group signature scheme can be integrated for the communication between the MU and the MK. By this way, the MK can also make sure that a request is coming from an authorized MU, without knowing the real identity of the corresponding MU.

Even though we encrypt the positions of the short reads (using OPE) to hide the conducted genetic tests from the biobank, the biobank might still infer the approximate positions of the short reads as a result of using OPE. The biobank does not see the exact bounds of the queries, but it can sort all short reads of the stored genome based on their offsets, which certainly gives it a rough idea which short read contains which nucleotides, and hence which genetic test is being performed. To avoid this, for each patient, we re-define the positions of the short reads before encrypting them using OPE (as discussed in detail in Section 7.1).

We also make sure that the MK cannot infer the genomic data of the patients by using the information it receives from the biobank and the cryptographic keys it stores. Indeed, as we will discuss in Section 7.2, we only provide the positions and the cigar strings (CSs) of a subset of the short reads (depending on the range of nucleotides requested by the MU) to the MK, which is not enough to infer the nucleotides residing in the contents of corresponding short reads (the contents of the short reads are never transferred to the MK). By only analyzing the CS (without having access to the content), the MK can learn the locations of some insertions and deletions in the patient's genome (but not the contents of these insertions or deletions). However, the MK cannot infer the locations or contents of the patient's privacy-sensitive point mutations (e.g., SNPs), which are typically used to evaluate the predispositions of the patients for various diseases. These privacy-sensitive point mutations can only be inferred when the CS is used together with the content of the short read (which is not revealed to the MK). Furthermore, as we mentioned in Section 4, by masking the encrypted short reads before providing them to the MU, we avoid the MU acquiring more genomic data than it requests.

Collusion between the parties (i.e., the biobank, the MK, and an MU) is not allowed in our threat model and we assume that laws could enforce this. Finally, all communication between the parties are encrypted to protect the system from an external attacker.

7 Privacy-Preserving Processing of Raw Genomic Data

7.1 Cryptographic Keys and Encryption of the Short Reads

As we discussed before, the position of a short read is in the form $L_{i,j} = \langle x_i | y_j \rangle$, where x_i represents the chromosome ($x_i \in [1, 23]$) and y_j represents the position of the short read on chromosome x_i ($y_j \in [1, 240\text{M}]$). Therefore, we represent the position of a short read as a 35-bit number, where the first 5 bits represent the chromosome number and the remaining 30 bits represent the position of the short read in the corresponding chromosome. If the positions of the short reads were

[7] Knowing the MU (e.g., the name of the hospital) the biobank could de-anonymize an individual using other sources (e.g., by associating the time of the test and the location of the MU with the location patterns of the victim). Thus, we hide the types of the conducted tests from the biobank to avoid it associating the conducted genetic test with the individual.

encrypted following this representation, the biobank could infer the approximate positions of the short reads as a result of using OPE (as discussed in Section 6).

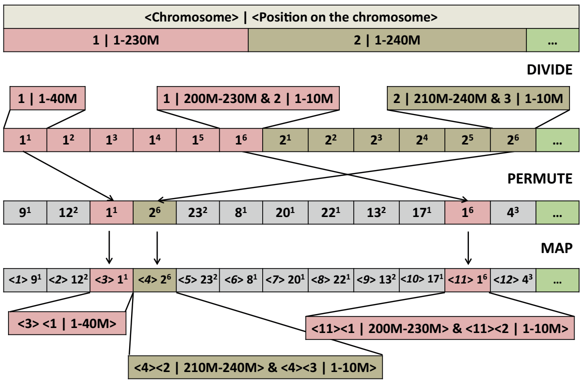
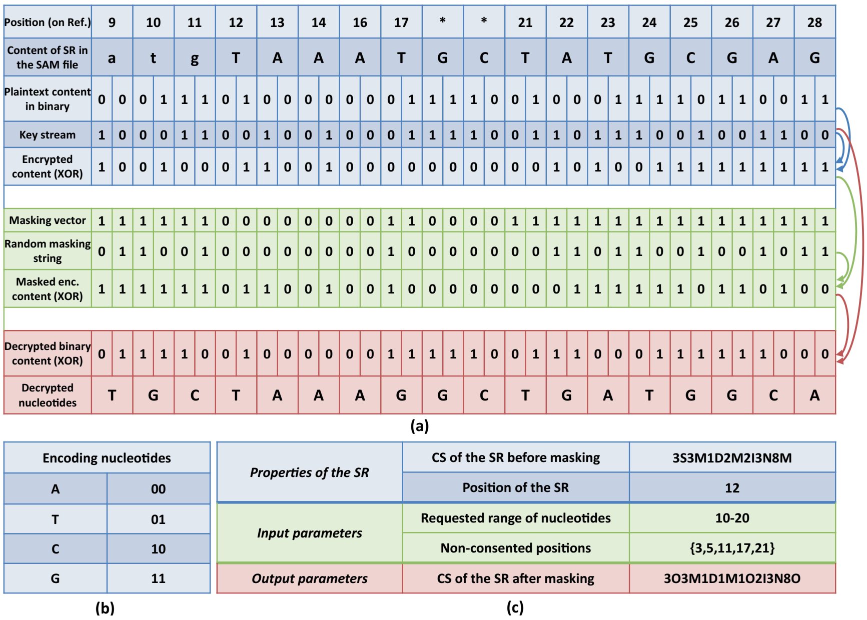


Fig. 3. Division, permutation and mapping of the positions on the whole genome.

To avoid this, we first divide the positions on the whole genome into parts of equal lengths, permute these parts, and then modify the positions in each part based on the permutation. In Fig. 3, we show such an example, in which the positions on the genome are divided into parts of length 40 million (totaling 75 parts as there are 3 billion nucleotides in the human genome). For example, chromosome 1 is divided into 6 parts $(1^1, 1^2, \ldots, 1^6)$, where the last part includes positions from both the first and second chromosomes. After division, all parts are permuted and mapped to different positions. As a result of the new mapping, the new position of a short read at $L_{i,j} = \langle x_i | y_j \rangle$ becomes $\mathfrak{M}(L_{i,j}) = \langle k \rangle \langle x_i | y_j \rangle$, where $\mathfrak{M}(.)$ is the mapping function for patient P, and k is the mapping of the corresponding part. For example, the position of a short read located in the first part of the first chromosome (part 1^1 in Fig. 3) becomes $\mathfrak{M}(L_{i,j}) = \langle 3 \rangle \langle x_i | y_j \rangle$ after the permutation and mapping. We note that as a result of the new positioning, we add κ bits (to represent the mapping) in front of the original positions of each short read ($\kappa = 7$ for the example in Fig. 3 as the positions on the genome are divided into 75 parts). Thus, for each patient, we re-define the positions of the short reads based on this new positioning, before encrypting the positions of the short reads using OPE. By doing so, we also change the ordering of the encrypted positions of the short reads. As a consequence, a curious party at the biobank cannot infer which part of the patient's genome is queried by the MU from the stored (encrypted) positions of the short reads. Finally, we assume that the MK keeps the mapping table $\mathfrak{M}_P$ (showing the mapping of each part in each chromosome) for each patient P. Note that as the permutation is done differently for each patient, the biobank cannot infer if two different patients are having a similar genetic test.

The different parts of each short read are encrypted as follows: (i) The positions of the short reads are encrypted using order preserving encryption (OPE), (ii) the cigar string (CS) of each short read is encrypted using a semantically secure symmetric encryption function (SE), and (iii) the content of each short read is encrypted using a stream cipher (SC). We note that an SC also provides semantic security, and although we really need an SC for the encryption of the content, one can also use an SC for the encryption of the CS.

We represent the key used for the semantically secure encryption scheme between two parties i and j as $K_{i,j}$. The symmetric OPE key that is used to encrypt the positions of the short reads of patient P is represented as K_P^O. Further, the master key of patient P, which is used to generate the keys of the SC is represented as M_P. We denote $K_P^{C_{i,j}}$ as the SC key used to encrypt the content of the



Position (on Ref.)	9		10		11		12		13		14		16		17		*		*		21		22		23		24		25		26		27		28	
Content of SR in the SAM file	a		t		g		T		A		A		A		T		G		C		T		A		T		G		C		G		A		G	
Plaintext content in binary	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	1	1	1	1	0	0	1	0	0	0	1	1	1	1	0	1	1	0	0	1	1
Key stream	1	0	0	0	1	1	0	0	1	0	0	1	0	0	0	1	1	1	1	0	0	1	1	0	1	1	1	0	0	1	0	0	1	1	0	0
Encrypted content (XOR)	1	0	0	1	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1	1	1	1	1	1	1	1
Masking vector	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Random masking string	0	1	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	0	1	0	0	1	0	1	1
Masked enc. content (XOR)	1	1	1	1	1	1	0	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	0	1	0	0
Decrypted binary content (XOR)	0	1	1	1	1	0	0	1	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0
Decrypted nucleotides	T		G		C		T		A		A		A		G		G		C		T		G		A		T		G		G		C		A	

(a)

Encoding nucleotides	
A	00
T	01
C	10
G	11

(b)

Properties of the SR	CS of the SR before masking	3S3M1D2M2I3N8M
	Position of the SR	12
Input parameters	Requested range of nucleotides	10-20
	Non-consented positions	{3,5,11,17,21}
Output parameters	CS of the SR after masking	3O3M1D1M1O2I3N8O

(c)

Fig. 4. Illustrative example for the encryption, masking and decryption of the content of a short read (SR). For the clarity of the example, we simplified the format of the short read position (which is in the form of $L_{i,j} = \langle x_i | y_j \rangle$, as discussed before). The arrows on the right show the inputs of the corresponding XOR operation. (a) Content of the SR (the 2 stars between positions 17 and 21 represent the positions at which the SR has insertions, G and C), its binary representation (following the encoding in (b)), the key stream to encrypt the corresponding content, and the format of the encrypted content (after the binary plaintext content is XOR-ed with the key stream). Furthermore, following the discussion in Section 7.2, we illustrate the masking vector generated at the MK considering the range of the requested nucleotides and the patient's consent (in (c)), the random masking string for the corresponding masking vector, and the format of the masked content (generated by XOR-ing the encrypted content with the random masking string). Finally, we show the format of the decrypted binary content, and the corresponding decrypted nucleotides. (b) Encoding format of the nucleotides A,T,C, and G. (c) Properties of the corresponding short read, requested range of nucleotides by the MU, non-consented nucleotides by the patient, and format of the CS after masking. We provide more details about different letters in the CS in [20].

short read whose position is $L_{i,j}$ (where $C_{i,j}$ represents the content of the short read with position $L_{i,j}$). We compute $K_P^{C_{i,j}} = \mathrm{H}(M_P, \mathcal{F}(L_{i,j}, S_{i,j}), L_{i,j})$, where $L_{i,j}$ is the (starting) position of the corresponding short read (on the DNA sequence), $S_{i,j}$ is a random salt to provide different keys for the short reads with the same positions, and H is a pseudorandom function. Moreover, $\mathcal{F}(L_{i,j}, S_{i,j})$ is a function that generates a *nonce* from the position and the random salt of the corresponding short read. We note that the random salts of the short reads are stored in plaintext. We represent the public-key encryption of message m under the public key of i as $\mathcal{E}(\mathcal{K}_i, m)$, the encryption of message m via a semantically secure symmetric encryption function (SE) using the symmetric key between i and j as $\mathrm{E}_{\mathrm{SE}}(K_{i,j}, m)$, and the OPE of message m using the OPE key of P as $\mathrm{E}_{\mathrm{OPE}}(K_P^O, m)$. Furthermore, we represent the SC encryption of the content of a short read as $\mathrm{E}_{\mathrm{SC}}(K_P^{C_{i,j}}, C_{i,j})$, where $C_{i,j}$ represents the content of the short read at $L_{i,j}$. In Fig. 4(a), we illustrate how the content of a short read is translated to plaintext bits and encrypted using SC (by XOR-ing the content with the key stream). Finally, in Fig. 5, we illustrate the format of an encrypted short read.[8]

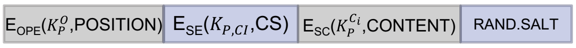


Fig. 5. Format of an encrypted short read. The size of each field is discussed in Section 9.

We assume that the certified institution (CI), where the patient's DNA is sequenced and analyzed, has K_P^O, M_P, and $K_{P,CI}$ ($K_{P,CI}$ is used to encrypt the CSs of the short reads) for the initial encryption of the patient's genomic data. These keys are then deleted from the CI after the sequencing, alignment, and encryption. We also assume that the patient's cryptographic keys for symmetric encryption, OPE, and SC are stored at the MK, and the patient does not participate in the protocol (except for giving his consent). Thus, for patient P, the MK stores K_P^O, M_P, and $K_{P,CI}$ along with the mapping table $\mathfrak{M}_P$ (as discussed before). Finally, the MU only stores the public key of the MK, $\mathcal{K}_{MK}$.

7.2 Proposed Protocol

Typically, a specialist at the MU (e.g., a physician at the hospital or a specialized clinic connected to the hospital) requests a range of nucleotides (on the DNA sequence of one or more patients) from the biobank (either for a personal genetic test or for clinical research). For simplicity of the presentation, we assume that the request is for a specific range of nucleotides of patient P. We note that when the MU is embodied in a pharmaceutical company, the MU does not know the real identities of the patients (i.e., participants of a clinical trial). Thus, in this case, the MU asks for a certain range of nucleotides of several pseudonymized patients from the biobank, who consented to participate in the corresponding clinical trial (the pseudonyms of these patients are known by the MU or by the biobank, and the general consent for the corresponding clinical trial is forwarded to the MK for masking). We illustrate the connections between the parties that are involved in the protocol in Fig. 6(a). In the following, we describe the steps of the proposed protocol (these steps are also illustrated in Fig. 6(b)).

- **Step 1:** The patient (P) provides a sample (e.g., his saliva) along with his permission to the certified institution (CI) for sequencing. We assume that laws prevent DNA sequencing of a (stolen) biological sample (e.g., hair) without the patient's permission.
- **Step 2:** The sample is sequenced by the CI. Next, the CI aligns the shorts reads of the patient with respect to the reference genome and constructs the SAM file of the patient. The short reads of the patient are also encrypted at the CI (as discussed in Section 7.1).
- **Step 3:** The CI sends the encrypted SAM file to the biobank along with the corresponding pseudonym of the patient. The CI also sends K_P^O, M_P, $K_{P,CI}$, and the mapping table $\mathfrak{M}_P$ for patient P directly to the MK via a secure channel (we do not illustrate this step in Fig. 6). We note that the first 3 steps of the protocol are executed only once.
- **Step 4:** A specialized sub-unit at the MU requests nucleotides from the range $[R_L, R_U]$ (R_L being the lower bound and R_U being the upper bound of the requested range) on the DNA sequence of patient P for a genetic test. We note that

[8] We discuss the size of each field (i.e., start and end positions of each field) in the encrypted short read in Section 9.

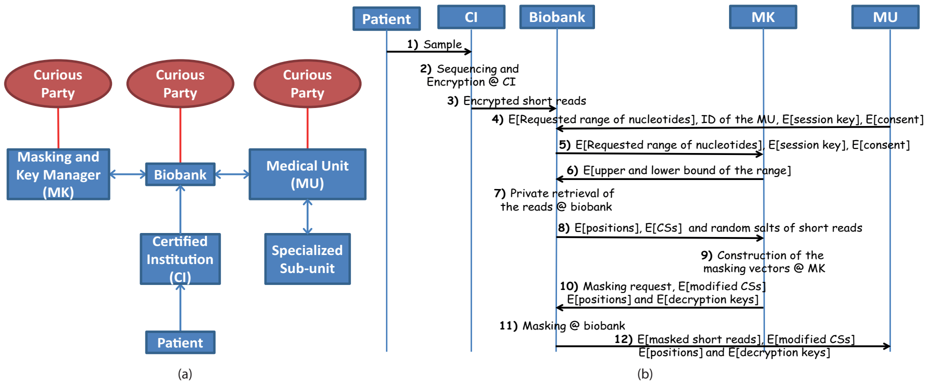

Curious Party
Curious Party
Curious Party
Masking and Key Manager (MK)
Biobank
Medical Unit (MU)
Certified Institution (CI)
Specialized Sub-unit
Patient
(a)
Patient
CI
Biobank
MK
MU
1) Sample
2) Sequencing and Encryption @ CI
3) Encrypted short reads
4) E[Requested range of nucleotides], ID of the MU, E[session key], E[consent]
5) E[Requested range of nucleotides], E[session key], E[consent]
6) E[upper and lower bound of the range]
7) Private retrieval of the reads @ biobank
8) E[positions], E[CSs] and random salts of short reads
9) Construction of the masking vectors @ MK
10) Masking request, E[modified CSs] E[positions] and E[decryption keys]
11) Masking @ biobank
12) E[masked short reads], E[modified CSs] E[positions] and E[decryption keys]
(b)

Fig. 6. (a) Connections between the parties in the proposed protocol. (b) The operations and message exchanges in the proposed protocol.

an access control unit stores the authorizations (i.e., access rights) of the original request owners (e.g., specialist at a hospital) to different parts of the genomic data. These access rights of different specialists to the SAM files are defined by the medical authorities. In our setting, the access control unit is the MU, and the MU checks the access rights of the original request owner before forwarding the request to the biobank. Once, the MU verifies that the original request owner has the sufficient access rights to the requested range of nucleotides, the MU generates a one-time session key $K_{MK,MU}$, which will be used for the secure communication between the MU and the MK (as we do not reveal the real identity of the MU to the MK, as discussed in Section 6, this key is generated for each session). The MU encrypts this session key with the public key of the MK to obtain $\mathcal{E}(\mathcal{K}_{MK}, K_{MK,MU})$.

The MU encrypts the lower and upper bounds of the requested range with $K_{MK,MU}$ to obtain $\mathrm{E}_{\mathrm{SE}}(K_{MK,MU}, R_L||R_U)$ and sends the corresponding request to the biobank along with the pseudonym of the patient P, the identification of the MU[9], $\mathcal{E}(\mathcal{K}_{MK}, K_{MK,MU})$, and $\mathrm{E}_{\mathrm{SE}}(K_{MK,MU}, \Omega_P)$, where Ω_P is the pseudonymized consent of the patient.[10] The MK uses this pseudonymized consent Ω_P to generate the masking vectors (as in Step 9).

- **Step 5:** Once the biobank verifies that request comes from a valid source[11], it forwards $\mathrm{E}_{\mathrm{SE}}(K_{MK,MU}, R_L||R_U)$, and $\mathrm{E}_{\mathrm{SE}}(K_{MK,MU}, \Omega_P)$, along with the pseudonym of the patient, and the encrypted session key $\mathcal{E}(\mathcal{K}_{MK}, K_{MK,MU})$ to the MK.
- **Step 6:** The MK decrypts the session key to obtain $K_{MK,MU}$ and decrypts the request ($\mathrm{E}_{\mathrm{SE}}(K_{MK,MU}, R_L||R_U)$) to obtain R_L and R_U. As we discussed before, the position of a short read is the position of the first aligned nucleotide in its content. Let Γ be the maximum number of nucleotides in a short read. Then, the short reads with position in $[R_L - \Gamma, R_L - 1]$ might also include nucleotides from the requested range ($[R_L, R_U]$) in their contents. Thus, the MK re-defines the lower bound of the request as $R_L - \Gamma$ in order to make sure that all the short reads (which include at least one nucleotide from the requested range of nucleotides) are

[9] We reveal the real identity of the MU to the biobank to make sure that the request comes from a valid source.

[10] Ω_P denotes the positions on the patient's genome for which the patient does not give consent to the original request owner (e.g., specialized sub-unit at the MU). Ω_P can be digitally signed by a medical authority to make sure that its content was not tampered with.

[11] We assume that the biobank has a list of valid MUs, whose requests it will answer.

retrieved by the biobank (as opposed to the lower bound, the MK does not need to re-define the upper bound of the request).

Next, the MK determines where $(R_L - \Gamma)$ and R_U are mapped to following the mapping table $\mathfrak{M}_P$ of patient P (as discussed in Section 7.1). If both $(R_L - \Gamma)$ and R_U are on the same part (e.g., in Fig. 3), then the MK computes the range of short read positions (to be retrieved by the biobank) as $[\mathfrak{M}(R_L - \Gamma), \mathfrak{M}(R_U)]$, where $\mathfrak{M}(.)$ is the mapping function for patient P. Otherwise (if they are not on the same part), due to the permutation of the parts (in Section 7.1), the MK generates multiple ranges of short read positions to make sure all short reads including at least one nucleotide from $[R_L, R_U]$ are retrieved by the biobank. For simplicity of the presentation, we assume $(R_L - \Gamma)$ and R_U are on the same part. Finally, the MK computes the encrypted range $[\mathrm{E}_{\mathrm{OPE}}(K_P^O, \mathfrak{M}(R_L - \Gamma)), \mathrm{E}_{\mathrm{OPE}}(K_P^O, \mathfrak{M}(R_U))]$, and sends this encrypted range to the biobank (with the pseudonym of P).

- **Step 7:** The biobank retrieves all the short reads (in the SAM file of patient P) whose encrypted positions $(\mathrm{E}_{\mathrm{OPE}}(K_P^O, \mathfrak{M}(L_{i,j})))$ are in $[\mathrm{E}_{\mathrm{OPE}}(K_P^O, \mathfrak{M}(R_L - \Gamma)), \mathrm{E}_{\mathrm{OPE}}(K_P^O, \mathfrak{M}(R_U))]$, and constructs the set $\Delta = \{\mathrm{E}_{\mathrm{OPE}}(K_P^O, \mathfrak{M}(L_{i,j})) : \mathrm{E}_{\mathrm{OPE}}(K_P^O, \mathfrak{M}(R_L - \Gamma)) \leq \mathrm{E}_{\mathrm{OPE}}(K_P^O, \mathfrak{M}(L_{i,j})) \leq \mathrm{E}_{\mathrm{OPE}}(K_P^O, \mathfrak{M}(R_U))\}$. As OPE preserves the numerical ordering of the plaintext positions, the biobank constructs the set Δ without accessing the plaintext positions of the short reads.

- **Step 8:** The biobank provides the encrypted positions in Δ along with the corresponding encrypted CSs and the random salt values of the short reads to the MK.

- **Step 9:** The MK decrypts the corresponding positions and the CSs of the retrieved short reads by using K_P^O and $K_{P,CI}$ in order to construct the masking vectors for the biobank. These masking vectors prevent the leakage of out-of-range content (in Fig. 1) and non-consented nucleotides (in Fig. 2) to the MU, as we discussed in Section 4.2.

The MK can determine the actual position of a short read from its mapped position as the MK has the mapping table $\mathfrak{M}_P$ for patient P (i.e., it can infer $L_{i,j}$ from $\mathfrak{M}(L_{i,j})$ using $\mathfrak{M}_P$). Using the position and the CS of a short read, the MK can determine the exact positions of the nucleotides in the content of a short read (but not the contents of the nucleotides, because the contents are encrypted and stored at the biobank). Using this information, the MK can determine the parts in the content of the short read that are out of the requested range $[R_L, R_U]$. Furthermore, the MK can also determine whether the short read includes any nucleotide positions for which the patient P does not give consent (the patient's pseudonymized consent, Ω_P, is provided to the MK in Step 5). Therefore, the MK constructs binary masking vectors indicating the positions in the contents of the short reads that are needed to be masked by the biobank before sending the retrieved short reads to the MU. We provide the details of the algorithm to construct the masking vectors in [20]. In Fig. 4(a), we illustrate how the masking vector is constructed for the corresponding short read, when the requested range of nucleotides is $[10, 20]$ and for a given set of nucleotide positions (on the DNA sequence) for which the patient P does not give consent (as in Fig. 4(c)).

The MK also modifies the CS of each short read (if it is marked for masking) according to the nucleotides to be masked. That is, the MK modifies the CS such that the masked nucleotides are represented with a new operation "O" in the CS.[12] By doing so, when the MU receives the short reads (which include the requested nucleotides), it can see which parts of them are masked (hence which parts of them it needs to discard for its purposes). In Fig. 4(c), we illustrate how the CS of the corresponding short read changes as a result of the masking vector in Fig. 4(a). Then, the MK generates the decryption keys for each short read (whose position is in Δ) by using the master key of the patient (M_P), positions of the shorts read, and the random salt values.[13]

• **Step 10:** The MK encrypts the positions, the (modified) CSs, and the generated decryption keys of the contents of the short reads, using $K_{MK,MU}$. Then, it sends the masking vectors along with the encrypted positions, CSs and decryption keys to the biobank. We note that in this step, the MK encrypts the actual positions of the short reads (e.g., $L_{i,j}$ instead of $\mathfrak{M}(L_{i,j})$) as these positions will be eventually decrypted and used by the MU, and the MU does not need to know the mapping table $\mathfrak{M}_P$ of the patient.

• **Step 11:** The biobank conducts the masking by XOR-ing the bits of the encrypted content of each short read (whose position is in Δ) with a random masking string. Each entry (bit) of the random masking string is assigned as follows: (i) If the corresponding entry is set for masking in the masking vector, it is assigned with a random binary value, and (ii) it is assigned with zero, otherwise. We provide the details of the algorithm to perform the masking at the biobank in [20]. Furthermore, in Fig. 4(a), we illustrate how the masked encrypted content for the corresponding short read is constructed by XOR-ing the random masking string with the encrypted content.

• **Step 12:** Finally, the biobank sends the encrypted positions, CSs and decryption keys (generated in Step 10 by the MK) along with the masked contents (generated in Step 11 by the biobank) to the MU. The MU decrypts the received data and obtains the requested nucleotides of the patient.

8 Evaluation

Focusing on the leakage of genomic data, we evaluate the proposed privacy-preserving system by using real genomic data to show (i) how the leakage of genomic data from the short reads threatens the genomic privacy of a patient, and (ii) how the proposed masking technique helps to prevent this leakage. We assume that the MU requests a specific range of nucleotides of patient P (e.g., for a genetic test) from the biobank. In practice, the requested range can include from one to thousands of nucleotides depending on the type of the genetic test.

First, without the masking in place, we observe the ratio of unauthorized genomic data (i.e., number of nucleotides provided to the MU that are out of the requested range) to the authorized data (i.e., number of nucleotides within the requested range) for various request sizes. For simplicity, we assume that all the nucleotides within the requested range are considered as consented data (i.e., the situation in Fig. 2 is not considered); and only those that are out of the requested range (but still provided to the MU via the short reads) are considered as the unauthorized data. For the patient's DNA profile (i.e., SAM file), we use a real

[12] Alternatively, the consent of the patient can be used by the MU instead of modifying the CS. Thus, the MU determines the masked nucleotides from the consent.

[13] The generation of the decryption keys for the SC is the same as the generation of the encryption keys as we discussed in Section 7.1.

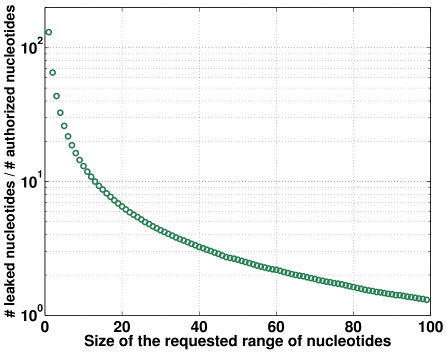


Fig. 7. Ratio of unauthorized (leaked) genomic data to the authorized data vs. the size of the requested range of nucleotides, when there is no masking in place.

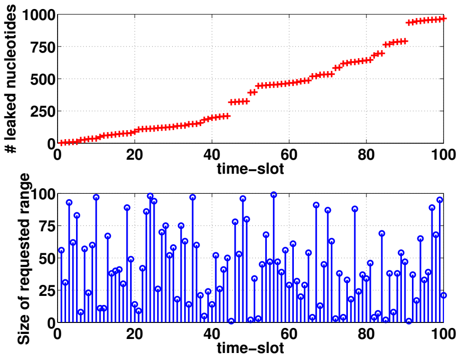


Fig. 8. Number of leaked nucleotides vs. time for various request sizes, when there is no masking in place.

human DNA profile [29] (with an average coverage of 8, meaning each nucleotide is present, on the average, in 8 short reads in the SAM file, and each short read includes at most 100 nucleotides) and we randomly choose the ranges of requested nucleotides from the entire genome of the patient. We illustrate our results in Fig. 7. We observe that for small request sizes, the amount of leakage (of unauthorized data) is very high compared to the size of authorized data. As the leakage vanishes (e.g., the ratio in Fig. 7 becomes 0) with the proposed masking technique, we do not show the leakage when the proposed masking technique is in place in Figs. 7-10.

Using the same DNA profile, we also observe the evolution in the amount of leaked genomic data over time. For simplicity of the presentation, we assume slotted time and that the MU conducts a genetic test on the patient at each time slot (by requesting a particular range of nucleotides from a random part of his genome). In Fig. 8, we illustrate the amount of genomic data (i.e., number of nucleotides) that is leaked to the MU in 100 time-slots. The jumps in the number of leaked nucleotides (at some time-slots) is due to the fact that some requests might retrieve more short reads comprised of more out-of-range nucleotides. As before, leakage becomes 0 when masking is in place, which shows the crucial role of the proposed scheme.

As discussed in Section 3, leakage of the nucleotides at the single nucleotide polymorphisms (SNPs) positions poses more risk for the genomic privacy of the patient. Therefore, we also study the information leakage, focusing on the leaked SNPs of the patient as a result of different sizes of requests (from random parts of the patient's genome). In Fig. 9, we illustrate the number of SNPs leaked to the MU in 100 time-slots. We observe that the number of leaked SNPs is more than twice the number of authorized SNPs (which are within the requested range of nucleotides). When the proposed masking technique is in place, the number of leaked SNPs (outside the requested range) becomes 0 in Fig. 9.

Finally, we study the genomic data leakage (number of leaked nucleotides and SNPs) when the MU tests the susceptibility of the patient [29] to a particular disease (i.e., when the MU asks for the set of SNPs of the patient that are used to test the corresponding disease). For this study, we use real disease markers [21]. We note that for this type of test, the size of the requested range of nucleotides (by the MU) for a single SNP is typically 1, but the SNPs are from several parts of the

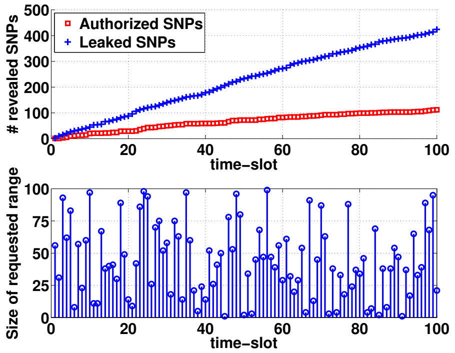


Fig. 9. Number of leaked SNPs vs. time for various request sizes, when there is no masking in place.

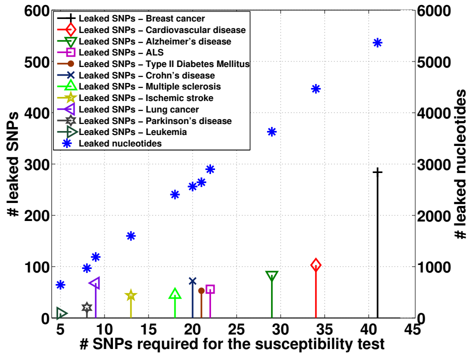


Fig. 10. Number of leaked SNPs and nucleotides during the susceptibility test to different diseases when there is no masking in place. The values on the right y-axis correspond to the number of leaked nucleotides.

patient's genome. In Fig. 10, we illustrate the genomic data leakage of the patient as a result of various disease susceptibility tests each requiring a different number of SNPs from different parts of the patient's genome (on the x-axis we illustrate the number of SNPs required for each test). We again observe that the leaked SNPs, as a result of different disease susceptibility tests, reveal privacy-sensitive data about the patient. For example, leaked SNPs of the patient as a result of a test for the Alzheimer's disease could leak information about the patient's susceptibility to "smoking behavior" or "diabetes" (in [20], we list the nature of some important leaked SNPs due to some susceptibility tests in Fig. 10). Similar to the previous cases, the number of leaked nucleotides and SNPs is 0 when masking is in place.

9 Implementation and Complexity Analysis

We implemented the proposed system and assessed its storage requirement and complexity on an Intel Core i7-2620M CPU with a 2.70 GHz processor under Windows 7. Our implementation is in Java and it relies on the MySQL 5.5 database. As before, for the patient's SAM file, we used a real DNA profile [29] including around 300 million short reads (each short read including at most 100 nucleotides) with a coverage of 8.

We used the Salsa20 stream cipher [30] for its efficiency and security. We also used the implementation of OPE from [31]. Finally, we used CCM mode of AES (with key size of 256-bits) for the secure communication between the MK and the MU by using the session key (in Section 7.2), and RSA (with key size of 2048-bits) for the public-key encryption (Step 4 in Section 7.2). We note that the security of the proposed scheme relies on the security of its underlying cryptographic protocols: (i) Salsa20 stream cipher [30] is proven to be a semantically secure encryption algorithm, (ii) the security of RSA relies on the problem of factoring large numbers and the RSA problem, and (iii) the security of OPE is recently analyzed by Popa *et al.* [23] to prove that the ciphertext values reveal no additional information about the plaintext values besides their order (i.e., IND-OCPA [31]).[14] As discussed, we

[14] Even though we used [31] for the implementation of OPE, a more recent version of OPE is shown to be more secure and faster [23]. We did not use the version of OPE in [23] due to the non-availability of a public implementation, but we are planning to integrate it in the future.

also prevent the security flaws (specific to genomic data) due to the knowledge of the orders of the encrypted positions by mapping the positions of the short reads to new values.

We structured the fields in the encrypted short read (in Fig. 5) as follows: We reserved the first 8-bytes for the encrypted position of the short read (via OPE). To save storage, we devoted the next 64-bytes of the encrypted short read to the CS and the content of the short read. As the input size of the stream cipher is 64-bytes, we encrypted the CS together with the content and other (header) information of the short read using the stream cipher. That is, out of the 64-byte input of the stream cipher, we allocated the first 20-bytes for the CS, the next 25-bytes for the content (as each short read in the used DNA profile includes at most 100 nucleotides), and the remaining 19-bytes for the remaining information about the short read (or padding). Finally, the last byte of the short read includes the plaintext random salt. Consequently, we computed the storage cost as 21.6 GB per patient. We note that stream cipher encryption does not increase the size of the data as it is the XOR of the key stream with the plaintext. The storage overhead (due to the proposed privacy-preserving scheme) is due to the encryption of the positions of the short reads by using OPE. A plaintext position is around 40 bits (depending the number of parts in Fig. 3) and an encrypted position is 8-bytes using the implementation of OPE in [31] (an encrypted position is 40-bytes using the more recent and secure version of OPE in [23]).

We also evaluated the computation times for different steps of the proposed scheme. The detailed computation times of different steps of the protocol can be found in [20]. Overall, it takes approximately 5 seconds for the MU to receive the requested range of nucleotides of the patient (Steps 4-12) after privacy-preserving retrieval and masking (for a range size of 100, which includes on the average 23 short reads), which shows the efficiency and practicality of the proposed scheme. We note that the computation time of the whole process is dominated by the retrieval of the reads at the biobank (which does not involve any cryptographic operations). Therefore, we can easily claim that the cost of cryptographic operations is not a bottleneck for the proposed protocol.

10 Conclusion

In this paper, we have introduced a privacy-preserving system for the storage, retrieval, and processing of aligned, raw genomic data (i.e., SAM files). The proposed scheme stores the SAM files of the patients at a biobank and lets the medical units (hospitals or pharmaceutical companies) privately retrieve the data (they are authorized for) from the biobank for genetic tests. We have shown that the proposed scheme efficiently prevents the leakage of genomic data and preserves the genomic privacy of the patients. We are confident that the proposed scheme will accelerate genomic research, because clinical-trial participants will be more willing to consent to the sequencing of their genomes if they are ensured that their genomic privacy is preserved.

11 Acknowledgements

We would like to thank Jurgi Camblong, Pierre Hutter, Zhenyu Xu, Wolfgang Huber, and Lars Steinmetz for their useful comments and suggestions.

References

1. E. Ayday, E. D. Cristofaro, G. Tsudik, and J. P. Hubaux, "The chills and thrills of whole genome sequencing," *arXiv:1306.1264*, 2013. `http://arxiv.org/abs/1306.1264`.
2. Presidential Commission for the Study of Bioethical Issues, Privacy and Progress in Whole Genome Sequencing (2012).
3. N. Homer, S. Szelinger, M. Redman, D. Duggan, and W. Tembe, "Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays," *PLoS Genetics*, vol. 4, Aug. 2008.
4. R. Wang, Y. F. Li, X. Wang, H. Tang, and X. Zhou, "Learning your identity and disease from research papers: Information leaks in genome wide association study," *CCS '09: Proceedings of the 16th ACM Conference on Computer and Communications Security*, pp. 534–544, 2009.
5. J. Gitschier, "Inferential genotyping of Y chromosomes in Latter-Day Saints founders and comparison to Utah samples in the HapMap project," *Am. J. Hum. Genet.*, vol. 84, pp. 251–258, 2009.
6. X. Zhou, B. Peng, Y. F. Li, Y. Chen, H. Tang, and X. Wang, "To release or not to release: Evaluating information leaks in aggregate human-genome data," *ESORICS'11: Proceedings of the 16th European Conference on Research in Computer Security*, pp. 607–627, 2011.
7. M. Gymrek, A. L. McGuire, D. Golan, E. Halperin, , and Y. Erlich, "Identifying personal genomes by surname inference," *Science: 339 (6117)*, Jan. 2013.
8. S. E. Fienberg, A. Slavkovic, and C. Uhler, "Privacy preserving GWAS data sharing," *Proceedings of the IEEE 11th International Conference on Data Mining Workshops (ICDMW)*, Dec. 2011.
9. J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik, "Privacy preserving error resilient DNA searching through oblivious automata," *CCS '07: Proceedings of the 14th ACM Conference on Computer and Communications Security*, 2007.
10. M. Blanton and M. Aliasgari, "Secure outsourcing of DNA searching via finite automata," *DBSec'10: Proceedings of the 24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy*, pp. 49–64, 2010.
11. S. Jha, L. Kruger, and V. Shmatikov, "Towards practical privacy for genomic computation," *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pp. 216–230, 2008.
12. F. Bruekers, S. Katzenbeisser, K. Kursawe, and P. Tuyls, "Privacy-preserving matching of DNA profiles," tech. rep., 2008.
13. P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik, "Countering GATTACA: Efficient and secure testing of fully-sequenced human genomes," *CCS '11: Proceedings of the 18th ACM Conference on Computer and Communications Security*, pp. 691–702, 2011.
14. M. Kantarcioglu, W. Jiang, Y. Liu, and B. Malin, "A cryptographic approach to securely share and query genomic sequences," *IEEE Transactions on Information Technology in Biomedicine*, vol. 12, no. 5, pp. 606–617, 2008.
15. E. Ayday, J. L. Raisaro, and J. P. Hubaux, "Privacy-enhancing technologies for medical tests using genomic data," *(short paper) in 20th Annual Network and Distributed System Security Symposium (NDSS)*, 2013.
16. E. Ayday, J. L. Raisaro, P. J. McLaren, J. Fellay, and J. P. Hubaux, "Privacy-preserving computation of disease risk by using genomic, clinical, and environmental data," *Proceedings of USENIX Security Workshop on Health Information Technologies (HealthTech)*, 2013.
17. E. Ayday, J. L. Raisaro, and J. P. Hubaux, "Personal use of the genomic data: Privacy vs. storage cost," *Proceedings of IEEE Global Communications Conference, Exhibition and Industry Forum (Globecom)*, 2013.
18. R. Wang, X. Wang, Z. Li, H. Tang, M. K. Reiter, and Z. Dong, "Privacy-preserving genomic computation through program specialization," *Proceedings of the 16th ACM Conference on Computer and Communications Security*, pp. 338–347, 2009.
19. Y. Chen, B. Peng, X. Wang, and H. Tang, "Large-scale privacy-preserving mapping of human genomic sequences on hybrid clouds," *NDSS'12: Proceeding of the 19th Network and Distributed System Security Symposium*, 2012.
20. E. Ayday, J. L. Raisaro, U. Hengartner, A. Molyneaux, and J. P. Hubaux, "Privacy-preserving processing of raw genomic data," *EPFL-REPORT-187573*, 2013. `https://infoscience.epfl.ch/record/187573`.
21. `http://www.eupedia.com/genetics/medical_dna_test.shtml`. Visited on 09/Apr/2013.
22. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pp. 563–574, 2004.

23. R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, 2013.
24. `http://articles.washingtonpost.com/2012-06-02/national/35462326_1_data-breaches-medical-data-social-security-numbers`. Visited on 09/Apr/2013.
25. D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," *Proceedings of the IEEE Symposium on Security and Privacy*, 2000.
26. S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," *Proceedings of the 2012 ACM Conference on Computer and Communications Security - CCS*, 2012.
27. O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious RAMs," *J. ACM*, vol. 43, pp. 431–473, May 1996.
28. E. Stefanov, E. Shi, and D. Song, "Towards practical oblivious RAM," *NDSS'12: Proceeding of the 19th Network and Distributed System Security Symposium*, 2012.
29. `ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data/NA06984/`. Visited on 09/Apr/2013.
30. D. J. Bernstein, "New stream cipher designs," ch. The Salsa20 Family of Stream Ciphers, pp. 84–97, Berlin, Heidelberg: Springer-Verlag, 2008.
31. R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting confidentiality with encrypted query processing," *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, 2011.