MaskAl: Privacy Preserving Masked Reads Alignment using Intel SGX

Christoph Lambert, Maria Fernandes, Jérémie Decouchant, and Paulo Esteves-Verissimo
SnT - Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg
christoph.lambert@uni.lu, jeremie.decouchant@uni.lu, maria.fernandes@uni.lu, paulo.verissimo@uni.lu

***Abstract*—The recent introduction of new DNA sequencing techniques caused the amount of processed and stored biological data to skyrocket. In order to process these vast amounts of data, bio-centers have been tempted to use low-cost public clouds. However, genomes are privacy sensitive, since they store personal information about their donors, such as their identity, disease risks, heredity and ethnic origin. The first critical DNA processing step that can be executed in a cloud, i.e., read alignment, consists in finding the location of the DNA sequences produced by a sequencing machine in the human genome. While recent developments aim at increasing performance, only few approaches address the need for fast and privacy preserving read alignment methods. This paper introduces MaskAl, a novel approach for read alignment. MaskAl combines a fast preprocessing step on raw genomic data — filtering and masking — with established algorithms to align sanitized reads, from which sensitive parts have been masked out, and refines the alignment score using the masked out information with Intel's software guard extensions (SGX). MaskAl is a highly competitive privacy-preserving read alignment software that can be massively parallelized with public clouds and emerging enclave clouds. Finally, MaskAl is nearly as accurate as plain-text approaches (more than 96% of aligned reads with MaskAl compared to 98% with BWA) and can process alignment workloads 87% faster than current privacy-preserving approaches while using less memory and network bandwidth.**

***Keywords*—Privacy, Read Alignment, Intel SGX**

I. Introduction

Deoxyribonucleic acid (DNA) is the molecule that encodes genetic information with an alphabet of four nucleotides: adenine (A), thymine (T), guanine (G), and cytosine (C). Thanks to the development of next-generation sequencing (NGS) technologies, DNA sequencing (i.e., the process of determining the order of nucleotides inside a DNA molecule) has been getting more affordable. For example, currently sequencing a human genome is estimated to cost around 1,000 US dollars. Moreover, with the associations between genomes and disease risks, phenotypes, or ethnicity, getting more precise, it is expected that studying genomes will enable progress in fields such as personalized and preventive medicine, genome editing, and forensics [1], [2]. Consequently, the production rate of genomic data has been greatly increasing all over the world [3].

During DNA sequencing, raw genomic data is acquired from a biological sample, and consists in millions of short DNA sequences (called reads) randomly sampled from a donor's genome. This raw data then go through a processing pipeline, whose final output is the donor's genome, a sequence of over 3 thousand million pairs of nucleotides. In the first step of this processing pipeline, the reads are aligned against a known reference genome, i.e., their position in the genome is identified. After the position of each read has been determined, scientists can identify the biological information the read contains through variant calling, which is the process of finding gene sequence variations between genomes.

However, processing and storing human sequenced data raises new privacy challenges [4]. Indeed, several privacy attacks have been described in the literature. Among them, the identification of a donor of publicly available Y-STR haplotypes information [5], and the discovery of a donor's predisposition to Alzheimer's disease [6]. Further risks include the possibility of leaked data to not only affect the donor in her lifetime, but also her relatives over generations. These attacks raised concerns about the possibility that leaked, or inferred, genomic data may be used to discriminate people in their everyday life [7] (e.g., to modify interest rates, deny health insurances, influence the selection of employees).

The access to critical private information (CPI) contained or linked to genomes is enabled thanks to an adversary using, or observing, one or several kinds of genomic variations: (i) short tandem repeats (STRs), where a short pattern is repeated a variable number of times; (ii) mutations (SNPs), insertions and deletions (indels), and copy number variations (CNVs); or (iii) disease-associated genes [8]. These CPI items represent up to 0.5% of a full genome, encode the differences between individuals [9], and can already be observed in raw genomic data. Therefore, in order to avoid data leakage and prevent privacy attacks, there is an urgent need to develop privacy preserving methods for DNA alignment, genomic data transfers and storage. In particular, currently used alignment approaches manipulate raw genomic data in plain-text and may run in public clouds. However, typically, clouds do not guarantee that the data cannot be accessed by either the cloud service provider, or an intruder [10]. Several works therefore highlighted the risks associated to the inconsiderate use of clouds for bio-medical data [11], [12].

To answer the need for privacy, secure alignment algorithms have been described in the literature. However, they sacrifice performance for security, and are much slower than plain-text algorithms [13], [14]. Garbled circuits alignment [15] and homomorphic encryption schemes [16] have been used to perform operations on encrypted reads. Recently, more practical solutions appeared. These solutions try to combine high performance and security by making controlled compromises on their security guarantees. Chen et al. [17] proposed a seed-and-extend alignment implementation for hybrid clouds. The initial ideas and limitations of this work inspired another solution, Balaur [18], which minimizes the use of a private cloud by relying on Locally Sensitive Hashing (LSH), secure

k-mer voting, and a MinHash algorithm. However, Balaur uses a lot of memory and requires heavy network transfers.

The challenge we address in this paper is the design of a lightweight alignment method that enforces the privacy of the genomic information contained in the reads, and yet can maintain a high performance by relying on widely available public clouds.

We propose MaskAl, a competitive approach for privacy-preserving read alignment using masked reads and Intel's Software Guard eXtensions (SGX). MaskAl is a two-tier hybrid system that uses an early stage filtering method to protect reads as soon as they are produced by the sequencing machines. The first tier (i.e., masked alignment) aligns masked reads in public clouds, and the second tier (i.e., score refinement) refines the results of the first tier by using an SGX implementation of the Smith-Waterman (SW) algorithm. Both steps can be massively parallelized on cheap public clouds (e.g., Amazon AWS) and trusted execution environments (e.g., Intel SGX). We therefore believe MaskAl to be an important step towards practical privacy enforcement during reads alignment, which is today routinely executed in insecure public clouds for performance and cost reasons.

An important key feature of our approach is the usage of SGX, Intel's newest Trusted Execution Environment (TEE), which protects applications against attack vectors heading from the operating system or hypervisor layer as well as attack vectors from the hardware layer. SGX uses a special hardware extension in the Memory Management Unit (MMU) called Memory Encryption Engine (MEE) for storing enclave code in a specific, cryptographically secured region in the RAM called Processor Reserved Memory (PRM) to ensure the same speed for SGX applications and non-SGX applications. Due to hardware limitations, the PRM has a maximum size of 128MB per Socket. In the literature, SGX has been used for privacy sensitive operations in the clouds, such as private web search [19], or MapReduce [20].

The main contribution of this work consists in the design of a privacy-preserving read alignment method, MaskAl, which uses SGX enclaves to securely refine the alignment scores in a public cloud, requiring access to the full reads. MaskAl is both *practical*, because it relies mostly on widely available public clouds, and *efficient*, since it is faster (no costly encryptions or network communications) and lighter (it only requires tens of GB of RAM) than the state-of-the-art methods. More specifically, the individual contributions of MaskAl can be detailed as follows:
1) We show how to efficiently partition reads in two sets, a privacy sensitive one and an insensitive one, based on the output of a state-of-the-art reads filter.
2) We design a hybrid cloud-based alignment scheme, which makes use of a public cloud, a private cloud, and an enclave cloud (i.e., a public cloud with TEEs).
3) We provide a performance and accuracy evaluation of MaskAl that shows its practicality.

The remainder of this paper is organized as follows: Section II details our system and threat models. Section III gives an overview of MaskAl, briefly explaining its processing steps and its architecture. Section IV recalls the principles of a read filter [21] that MaskAl uses to create insensitive and sensitive versions of reads, depending on whether or not they carry genomic variations. Section V-B describes MaskAl's approach in detail, which uses sensitive and insensitive versions in a privacy preserving and efficient way. Section VI details the read alignment verification and refinement that MaskAl executes in SGX enclaves. Section VII provides the performance evaluation we ran on MaskAl and on state-of-the-art alignment algorithms. Section VIII provides an overview of the related works. Finally, Section IX concludes this paper.

II. System and Threat Models

A. System model

We consider a scenario where a bio-center receives a biological sample, e.g., in the form of blood or saliva probes, from a donor, and uses one of its sequencing machines to digitize the DNA it contains. We assume that this bio-center has limited computing resources, and wants to rely on public clouds to scale for economical reasons. Therefore, the system architecture we consider makes use of public, private and enclave clouds to enforce the privacy of the sensitive data contained in reads, and maintain a high throughput.

Public clouds are server farms which can be rented, like Amazon AWS, or Google Cloud. These machines are very cheap and everything related to them can be or is managed by outsourced third-parties (maintaining the systems, operational tasks or even development tasks). Typically, public clouds do not guarantee that the data will never be accessed by either the cloud service provider, or an intruder [10]. In particular, in the bio-medical field, several works highlighted the risks associated to the inconsiderate use of public clouds for bio-medical data [11], [12], [22], [23].

Enclave clouds are SGX-powered public clouds that guarantee secure data operations inside the enclaves. We assume the costs for an hour of operation in public clouds and enclave clouds to be equally low, as newer Intel CPUs usually support SGX. Given the technical restriction of 128MB protected per SGX CPU, it could be possible that more SGX nodes are needed for parallelization, as multi-threading on one CPU would divide the usable PRM by the number of threads. Due to the technical restrictions neither the full alignment nor the filtering step can suitably be done inside an enclave today. However, the used SGX based enclaves could be replaced by any other suitable TEE.

Private clouds contain servers hosted by the bio-company or the research center who own the sequencing machines. The owner of these machines has full control over both the hardware (i.e., physical and virtual) and the software stack of these machines, to enforce the highest possible security level. Maintaining and using private clouds is obviously more expensive than using the two previous kinds of clouds we presented, which encourages bio-companies and research centers to offload their computations to public clouds. Filtering will take place here – since filtering is a fast and very inexpensive operation compared to a full alignment in private clouds.

B. Threat model

Privacy and raw reads. We consider an honest-but-curious adversary located in the cloud whose goal is to infer

privacy-sensitive information about the donor whose genome is sequenced. This adversary would first rely on raw genomic sequences it may be able to observe during the alignment of a subject's reads. If the adversary is able to align those reads, as the legitimate workflow is trying to, and identify the genomic variations they carry, a trail attack becomes possible [24], [25]. In such an attack, DNA samples are matched to their identified subject through the use of unique distinguishing features in different institutions where the subject leaves traces of his genome. For example, if a donor later participates in a public bio-medical study, the adversary may be able to identify the subject in the results based on her known genomic characteristics, and infer private information (e.g., that the donor suffers from a disease associated with the study). Read alignment is therefore the first stage in the genomic pipeline where privacy-sensitive genomic features can be observed by an adversary, if they are not protected [10].

Trusted execution environment (TEE). We assume that the TEEs we use for secure computation in clouds, i.e., Intel SGX enclaves, are secure. As [26], [27], [28] showed recently, it is possible with common side channel attacks like PRIME+PROBE to leak confidential information from the cache. However, mitigation methods for the mentioned side channel attacks discussed in [26], [28] can be used by the enclave developers.

Cryptography and enclave initialization. We store all data needed by the Smith-Waterman algorithm (i.e., the reference genome and the full reads) encrypted on the enclave clouds, to prevent any inference attack based on the accesses an enclave would perform on these data structures. More specifically, we rely on AES128-GSM as it is integrated in the SGX SDK to encrypt the plain-text reads and the reference genome frames, and store them in the main memory of the machine hosting the enclave. In addition, we encrypt communications between the various types of clouds using RSA2048, and assume that the adversary is not able to decrypt encrypted communications. Deploying an enclave would successively consist in its remote attestation and its initialization with the encrypted data structure, sent over the network.

III. Masked Alignment Overview

Given our threat and system models, we propose a modified read alignment workflow based on the partitioning of reads at the nucleotide level. Right after it has been generated by a sequencing machine, the location of a read in the human genome is unknown. It is only after its alignment that a read's location is known, and therefore whether or not it carries sensitive information. We rely on a state of the art read filter to identify the nucleotides that are potentially sensitive.

The main idea behind our approach is to use masked reads to perform privacy-preserving read alignment on public clouds with existing state-of-the-art alignment algorithms to avoid much more expensive private clouds. Indeed, as we show later (see Figure 3), it is possible to align masked reads in public clouds, in a privacy-preserving manner, using existing alignment algorithms with a true positive rate of at least 96%.

Figure 1 provides an overview of MaskAl's alignment scheme. We detail its main steps in the following.

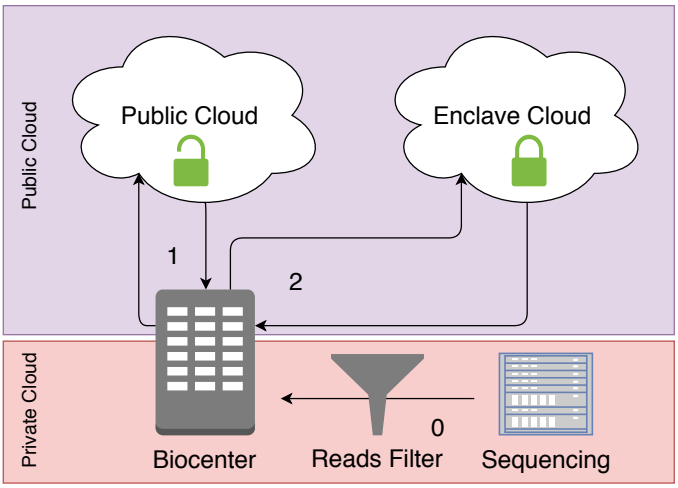

Fig. 1. Overview of MaskAl's network flows: (0) filtering of raw reads (FASTA or FASTQ files); (1) alignment of masked reads (SAM files) in a public cloud and transmission of the results back to the bio-center; (2) encrypted transmission of the reads to an enclave and refinement of the alignment scores

Step 0 [Reads filtering and masking] After reads have been sequenced, the bio-center associates to each of them a header that contains its anonymous donor's ID and its read ID. Reads are then filtered using the read filter we introduce in Section IV. During the filtering process, the nucleotides of each read are either classified as sensitive or insensitive. Reads are then partitioned in two versions, one which contains only the sensitive nucleotides, and the other one which contains only insensitive nucleotides and hide the length of sensitive sections. Indeed, since genomic variations can have different lengths, the length of sensitive sequences in the insensitive version of a read is hidden (collapsing of the hiding symbols), so that the insensitive version of a read never reveals any information. The filtering and the masking of reads can take place either in the sequencing machine or in a private cloud directly after sequencing.

Step 1 [Masked reads alignment] The sensitive version of reads remains securely stored inside the private cloud, while the insensitive version of reads is sent to the public cloud. We use cryptographic mechanisms to ensure the integrity of the reads. The alignment of the insensitive versions of the reads is performed in the public cloud. The original reads received by the public cloud from the bio-center are returned to the bio-center, along with the candidate alignment position that the public cloud obtained. The data transferred for the masked alignment is limited to a file containing the masked reads (in a FASTQ or FASTA file format) and a file that contains the alignment results (in the SAM file format).

Remark 1. Depending on the reads properties, the alignment process may be stopped here, namely if the vast majority of the reads have been correctly aligned to the reference genome. Otherwise, the process continues with step 2. In this case, the processing could be optimized by having the public cloud sending the candidate positions directly to the enclave cloud. We do not present this option here for simplicity.

Step 2 [Alignment score refinement] A second step,

performed in enclaves, may be needed for two reasons: (i) it is not possible to compute a reliable alignment score based only on the masked version of reads; and (ii) the few reads that have not been correctly aligned during masked alignment can be identified, and if necessary tentatively realigned in a secure environment. The bio-center sends the encrypted full reads, and their candidate alignment positions obtained during Step 1, to the enclave cloud. The enclaves validate and refine the possible candidate positions for each read using the full read's unmasked information. Since the enclave cloud is a trusted environment, the alignment score refinement can use the unmasked (plain-text) read instead of the masked representation. Besides refining the alignment positions of reads, this step also detects reads whose candidate positions are incorrect. Reads may have been incorrectly aligned by the public cloud because they contain too many sequencing errors, or because they contain too many sensitive nucleotides. After obtaining the results of the alignment score refinement, the enclave replies to the bio-center with the read IDs and their alignment position, and alignment score.

Remark 2. As we show in Figure 3, the proportion of misaligned reads is small enough to be ignored. Indeed, variant calling, the processing step that follows read alignment in the genomic workflow, can tolerate some unaligned or misaligned reads. Otherwise, if one would absolutely want to realign those reads, they would have to be aligned inside the private cloud in plain-text, or using state-of-the-art secure algorithms in the public cloud.

IV. Reads Filtering and Masking

In this section, we briefly describe a recent filtering method [21] that detects sensitive nucleotides in genomic sequences of arbitrary lengths, which we use in MaskAl to create a privacy insensitive (i.e., that does not carry known genomic variations), as well as complementary sensitive version of the reads. In the following, we recall the main ideas the long reads filter is built on, and refer to the full article for the implementation and evaluation details.

The long reads filtering (LRF) approach can be described in three main steps: dictionaries generation, Bloom filters initialization, and reads filtering. LRF receives as input a read, directly at the mouth of a sequencing machine, and outputs two complementary reads. The first one contains only letters that are not part of genomic variations, and therefore can be manipulated in plain-text in public clouds, while the second one contains letters which are suspected to be part of a genomic variation, and therefore has to be transmitted encrypted, or manipulated inside an enclave, according to our threat model.

Dictionaries generation consists of collecting all the known variations in the 1000 Genomes Project, and reconstructing their neighboring regions using the nucleotides from the reference human genome used to describe them. Since individuals present combinations of several genomic variations, located at different locations, the LRF then enumerates all theoretically possible combinations of the collected genomic variations. Then, the LRF creates sequences of $K = 33$ nucleotides from these combinations to build the dictionaries, where $dict_i$ denotes the dictionary that contain sequences of K nucleotides whose i^{th} nucleotide is sensitive ((K, i)-sensitive sequences).

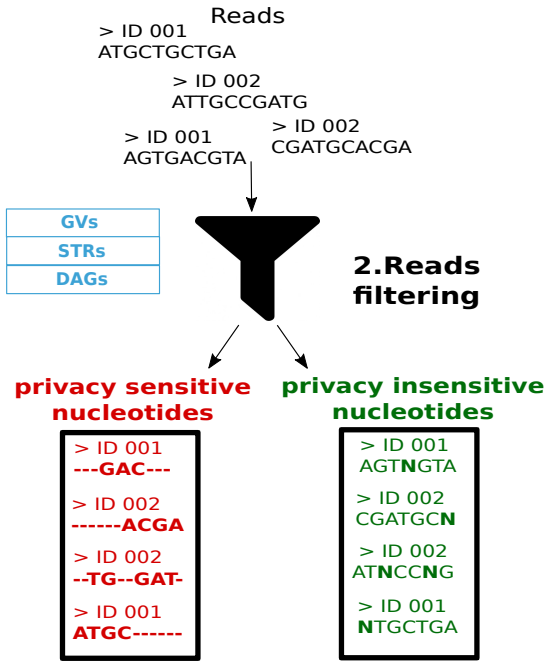

Fig. 2. Infrastructure and workflow overview. Phase 1 represents the sequencing of a genome, which produces DNA reads. Phase 2 represents the detection of the sensitive nucleotides in reads based on the read filter, and the partitioning of reads according to their sensitive and insensitive versions.

The long read filter creates one or several Bloom filters [29], [30], and initializes them from the previously generated dictionaries of (K, i)-sensitive sequences. To tolerate sequencing errors the long read filter relies on several dictionaries, using one Bloom filter per dictionary.

Finally, filtering the reads consists in moving a sliding window of size K over them and testing whether all subsequences have been inserted in one of the Bloom filters. More precisely, when the sequence contained in one of the studied windows matches one Bloom filter, say the one initialized with the (K, i)-sensitive sequences, then its i^{th} nucleotide is tagged sensitive. This way, for each detection of a sequence in a Bloom filter, only one nucleotide is detected sensitive.

Depending on the positions of the nucleotides in the sequences that the filter is able to detect as sensitive, some nucleotides either at the end, or at the beginning, of a read, may not be detected sensitive. These nucleotides are preventively treated sensitive (masked). However, this effect is attenuated once several Bloom filters are used, initialized with different sensitive positions. Finally, the evaluation of LRF shows that, given known privacy attacks, the privacy of donors is protected, since attackers are not given access to enough privacy sensitive information (around 7 SNPs per genome).

V. Masked Reads Alignment

A. Read partitioning

The output of the filter consists of the input read along with a set of the nucleotides flagged as sensitive. Revealing the insensitive nucleotides of a subject does not reveal any distinctive information, since by definition they are common to all subjects.

After the nucleotides of a read have been either tagged as sensitive or insensitive, MaskAl then divides a read into two complementary parts, which can be reunited if needed. More specifically, in the insensitive version of a read, we replace regions of consecutive sensitive letters by an 'N' (i.e., the green file in Figure 2), so that no information is revealed about the hidden section, and all privacy insensitive letters are replaced by a '-' in the privacy sensitive version of a read (i.e., the red file in Figure 2). In practice, MaskAl keeps only the full version of a read, and its privacy insensitive version.

B. Masked reads alignment

Our first design choice consisted in selecting a plain-text alignment algorithm, among the many that have been developed recently [31], [14], [32], [13], [33] to align masked reads to the reference genome in public clouds.

To do so, we considered two representatives of two different algorithm families: (1) LAST [33] as a representative of hashmap based algorithms; and (2) BWA [13] as a representative of suffix tree based algorithms. We chose LAST because it has the ability to find weak matches with big gaps better than other algorithms in its category, which we assumed would help given the nucleotides that MaskAl masks in the reads. According to its high performance in both multi- and single-threaded scenarios we chose BWA as a representative for FM-index based algorithms. In practice, we found out that BWA has a higher performance and better accuracy than LAST, even with masked reads.

We performed alignment experiments on simulated masked reads using LAST and BWA, and measured their accuracy, but also their performance and memory footprint.

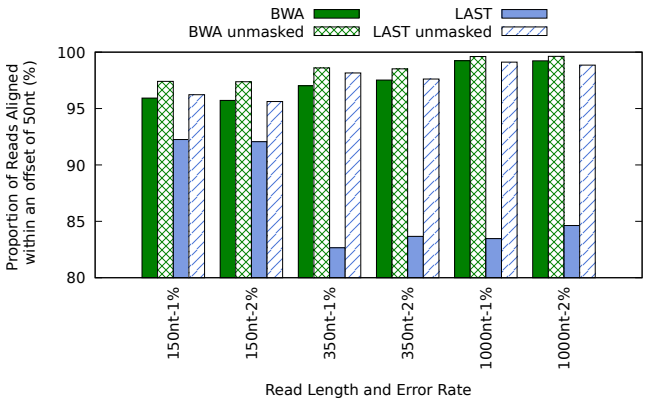

Fig. 3. Comparison of the alignment precision using masked and unmasked reads with BWA and LAST.

Figure 3 presents the proportion of the input reads that have been correctly aligned when applying BWA and LAST. Experiments have been performed on both masked and unmasked versions of the reads, to observe the impact of masking reads. Without masking, BWA is able to align between 98% and 99.9% correctly, while LAST can align between 95% and 99%. However, BWA aligns masked reads significantly more accurately than LAST (e.g., with reads of 1000 nucleotides and 2% error rate LAST aligns 84% of the reads and BWA 99%). In addition, the alignment accuracy of BWA on masked reads is always higher than 96%, which proves that aligning masked reads provides a good accuracy.

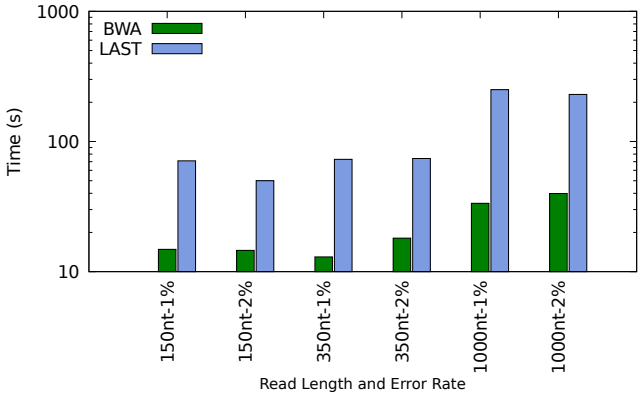

Fig. 4. Alignment times for masked reads using BWA and LAST.

In addition to a higher accuracy, we observed that BWA is also faster than LAST. Figure 4 shows that, on the same sets of masked reads, BWA is always at least one order of magnitude faster than LAST. Moreover, BWA uses special data structures (FM-index) which need less resources (memory and time) than LAST.

After those experiments, it makes sense to rely on BWA to align masked reads. The output of BWA on masked reads is made of candidate alignment positions. Indeed, BWA cuts a read into several sections, delimited by the N characters we introduced during the masking of reads, which it individually tries to align.

VI. Alignment Score Refinement in SGX Enclaves

In this section, we first present the final step of the reads alignment, which consists in refining the alignment scores around the candidate locations found during the alignment of the masked reads. This alignment refinement is executed in SGX enclaves for privacy, since the full version of the reads has to be used. Then, we recall how the algorithm we implemented in enclaves works — namely the Smith-Waterman algorithm — and explain how we adapted it so that it would run inside enclaves using limited memory.

A. Score refinement and alignment extension

After several candidate positions have been identified for a read, MaskAl verifies which one is the most accurate by comparing the region around the candidate positions with the read. In practice, the overhead of this verification is reasonable, since the alignment verification is computed in a small region around the alignment candidate matching positions. This verification phase is similar to the extend phase of many alignment algorithms, and therefore could replace it in those algorithms. In practice, we chose not to execute both the masked alignment and the extension in enclaves on the same machines, to eliminate attacks on correlated encrypted messages (i.e., the masked reads and the encrypted reads).

Since this step requires the full reads, we execute it inside an Intel SGX enclave. Compared to other TEEs like ARM

TrustZone, we chose to use SGX since Intel ships it with out-of-the-box abilities like local and remote attestation, and with a user friendly development environment. Those features can be criticized, as the infrastructure provided by Intel poses a single point of failure and an attack vector for very sophisticated attackers. However, note that this is just one prototype option for our method. In fact, the enclave cloud we use could be replaced by any trusted execution cloud environment.

B. The Smith-Waterman algorithm

The Smith-Waterman algorithm (SW) [34], [35] is used to estimate the local alignment of two strings. SW compares segments of all possible lengths with an edit-distance-based similarity measurement. In particular, the two strings are compared character by character, and matches, mismatches and deletions are weighted differently. The local alignment estimates the longest most similar fragment of two strings.

To calculate the local alignment, SW uses a computing matrix M (see Table I). Row 0 of M contains the reference string and column 0 contains the sequence to align. Both strings have an empty character added in front of the first letter. To initialize the computing matrix M, each element of M is initialized with 0. Besides the computing matrix M there is also a so called traceback matrix T, which helps to reassemble the correct alignment after computing M. T is initially a copy of initialized matrix M. In addition to the four edit-distance score cases (insertion, deletion, match and mismatch), SW introduces a fourth case: 0. The value of an element M(n,m) is computed according to Figure 5.

	ε	A	C	G	T	A
ε	0	0	0	0	0	0
A	0	2	1	0	0	2
G	0	0	0	3	2	1
T	0	0	0	2	4	3
C	0	0	2	1	3	3

	ε	A	C	G	T	A
ε	0	0	0	0	0	0
A	0	D	L	0	0	0
G	0	0	0	D	0	0
T	0	0	0	0	D	0
C	0	0	0	0	0	0

TABLE I. SMITH-WATERMAN SCORE MATRIX (LEFT) AND TRACEBACK MATRIX (RIGHT). THE WEIGHTS USED ARE: MATCH/MISMATCH 2/-1 AND GAP PENALTY: -1. THE ALIGNMENT RESULT ACCORDING TO BOTH TABLES IS "ACGT" AND "A-GT"

To get the correct local alignment, SW fills a second matrix T, which stores the solution directions. An element in the backtracking matrix T is either D (diagonal), L (left) or U (upper). D, L and U are the directions, to find the predecessor of an element. If M(i,j) was chosen because the match-mismatch case gave the maximum value, T(i,j) equals D, the diagonal direction. It is called diagonal, because the match-mismatch computation uses the value of M(i-1, j-1), and position (i-1,j-1) is in fact the upper diagonal to (i,j). Following the same pattern, insertion equals U and deletion equals L, because insertion uses the upper neighbor whereas deletion uses the left neighbor to compute the element (see Figure 5). To find the actual alignment result, backtracking starts at the position of the element in M with the highest value. Starting from there, backtracking takes the same way backwards by parsing the directions of the backtracking matrix. Smith-Waterman's backtracking terminates whenever the value of a reached element in the backtracking matrix is 0. This guarantees that only the best fitting fragment of a sequence is computed. A comprehensive example of the Smith-Waterman algorithm is shown in Table I.

$$Score_{i,j} = max \begin{cases} 0 & \\ Score_{i-1,j-1} & + \quad 2 \quad : u = v \\ Score_{i-1,j-1} & +(-1) \quad : u \neq v \\ Score_{i,j-1} & +(-1) \quad : insertion \\ Score_{i-1,j} & +(-1) \quad : deletion \end{cases}$$

Fig. 5. Scoring function for Smith-Waterman algorithm example: u,v are the characters of the two strings which are currently compared

C. Running the SW algorithm in SGX enclaves

To perform the Smith-Waterman algorithm for each read in an enclave, an adequate reference frame is needed. Traditionally, the Smith-Waterman algorithm is performed with a read and the complete reference DNA (i.e., a sequence of 3 Gigabases). Fortunately, by aligning a masked read with BWA, the result will be, in at least 96% of the cases, a position within an offset of 50 nucleotides. Therefore, we use a shortened reference for our enclave based SW algorithm. The shortened reference is cut starting 50 nucleotides (nt) before the actual candidate position (candidate position - 50) with the length of the aligned read plus 50nt.

Running the Smith-Waterman algorithm in SGX enclaves present some difficulty, since enclaves have a limited memory (128MB in total ca. 90MB usable) that the algorithm tends to exhaust very quickly. We circumvented this problem by assuming the maximum length of the reads to be aligned to 1000 nucleotides. We chose a maximum length of 1000 because of two reasons: firstly, 1000 nucleotides easily fit into an enclave, so that an enclave could handle two threads in parallel; and secondly, many of current sequencing techniques are producing reads of less than 1000 nucleotides. This current limitation of our prototype may disappear in the future, if future versions of SGX extend or even remove their current memory limitation.

To reduce the network traffic we assume that the reference resides in encrypted overlapping chunks of 2000 nucleotides on each enclave. We use overlapping segments of the reference genome to avoid having a read located on two chunks, which would incur more encryptions and decryptions for the enclaves. To execute the validation, the enclave application uses an hashmap to find the correct encrypted reference slice and decrypts it, together with the encrypted unmasked read into the enclave's Processor Reserved Memory (PRM). Then, SW is executed with the shortened reference and the unmasked read. After the score refinement step, the requesting bio-center receives the alignment result encrypted.

VII. PERFORMANCE EVALUATION

A. Experimental settings

Data sources. We rely on the genomic variations and the individual genomes from the 1000 Genomes Project [36], along with the short tandem repeats from the Tandem Repeats Database [37] to create the dictionaries of sensitive sequences, and filter individual genomes. We use the Phase 3 20130502 release of the 1000 Genomes Project, and the associated GRCh37 reference genome against which we align reads.

Hardware. For our experiments we used three environments, which correspond to the public, private and enclave machines. The first machine, which corresponds to the private cloud, is a quad socket Intel Xeon E5-4650 v3 processor with 12 cores running at 2.10 GHz. We used this first machine to perform the filtering step used in MaskAl. This machine is equipped with 190 GB of RAM, and has 15 TB of disk storage. The second machine, which represents the public cloud, is an HPC shared machine equipped with Intel Xeon E7-4850 processors where we used 30 threads for our tests. This machine is equipped with 1 TB RAM and disposes of up to 150 GB of hard disk. We relied on this machine for the experiments with Balaur, which required more memory than our local machine possesses. We emulated the enclave cloud with a desktop machine containing Intel i7-6700 processors. We consider a common network bandwidth of 1Gbit/s between the private cloud, where the sequencing machine is hosted, and the public cloud, which contains both enclave-less and enclave-equipped processors.

Software. All parts of our prototype are written in C/C++. We used the official Intel SGX SDK in version 1.9 and run the code on Ubuntu (16.04 LTS). To run Smith-Waterman securely in the enclave, we created an ECALL which takes the encrypted reads as parameter, decrypts them inside the PRM and run Smith-Waterman with the plain data inside the enclave. We assumed the availability of a working secure remote attestation infrastructure and transport layer encryption. Since alignment is a massively parallelized application, we expect MaskAl to scale linearly with the number of enclaves used, and therefore study its performance when running on a single thread.

Simulated reads. We perform experiments on simulated reads that we generate using wgsim [38] and the GRCh37 reference genome. We used the simulated reads to evaluate the alignment algorithms with a wide range of realistic read parameters (e.g. reads length, error rate) to mimic reads from different sequencing technologies. We follow the characteristics of existing sequencing technologies to combine these parameters. The lengths of the reads span from 150 to 1000 nucleotides, while we consider error rates of 1% and 2%. In each category, we simulated 100000 reads from the entire human genome.

Selected alignment algorithms. Among the existing plain-text alignment algorithms, we selected one algorithm for each category of alignment paradigm we identified. The comparison between BWA and LAST algorithms is summarized in section V-B. Due to its faster alignment time, and higher accuracy with masked reads, we selected BWA for our performance evaluation. We also compare with existing hybrid algorithms, using the recent solution developed by Popic et al. [18], Balaur. Balaur is a privacy-preserving alignment algorithm designed for hybrid clouds, which uses locality sensitivity hashing and kmer voting. This approach uses data encryption to securely outsource some computations to the public cloud. Alignment algorithms proven to be secure have been presented in the literature. Existing solutions are either based on garbled-circuits [15], or on homomorphic encryption schemes [39], [40]. However, since those algorithms are not practical enough, because of their low throughput, we do not compare MaskAl against those alignment methods.

Criteria for the evaluation of alignment algorithms. Our first goal is to show that the masking we operate on reads, using the reads filter, does not significantly degrade the accuracy of the read alignments. Then, we monitor several performance metrics, such as the memory consumption, the amount of network communications, and the computational overhead of the alignment algorithms. Previous work showed that providing high alignment performance is critical for an alignment algorithm to be adopted by the bio-medical community [41].

B. Memory consumption

Balaur requires the indexing of the reference genome, which takes as a parameter the length of the reads to be aligned (150, 350, and 1000 nucleotides). This process requires from 62GB to more than 189GB of memory, depending on the length of the sequences to align (longer sequences require less memory). In contrast, MaskAl only needs tens of GB of memory, which are used to filter and mask the reads.

The amount of RAM needed during the alignment is the next important parameter for choosing an adequate public cloud. While Balaur needs between 67GB to 153GB of RAM, BWA requires between 5.2GB and 5.9GB, a considerable lower memory requirement (see Figure 6).

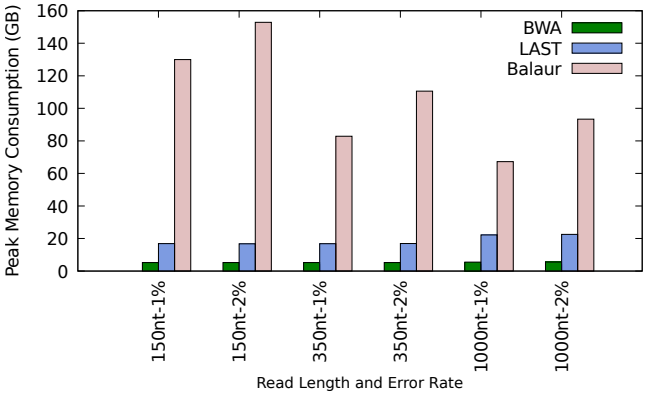

Fig. 6. Comparison peak memory usage of BWA, LAST and Balaur

The peak memory consumption for the score refinement in the enclave is limited to the 128MB sized Processor Reserved Memory (PRM). As we did not artificially increase the size of the PRM by using a SWAP-like technique, due to performance issues, the 128MB PRM limit is the current upper bound for MaskAl's enclave validation.

C. Alignment time - masked alignment

MaskAl's computation time can be splitted into three parts: (i) the masked alignment; (ii) the enclave validation (score refinement); and (iii) the time required for transfers between the bio-center, the public cloud and the enclave cloud.

Starting with the first step – the masked alignment – we experienced a big difference between LAST and BWA. LAST consumes more time the longer the reads are and the higher their error rate is. BWA needs in average 80% of the time taken by LAST for the same task with the same amount of threads. BWA computation time slightly increases for longer reads. Compared to LAST, BWA computation time is less influenced by high error rates (see Figure 4). Section V-B presented our results for this phase of the alignment process, and based on

those results, we chose to rely on BWA to align the masked reads.

D. Alignment time - score refinement

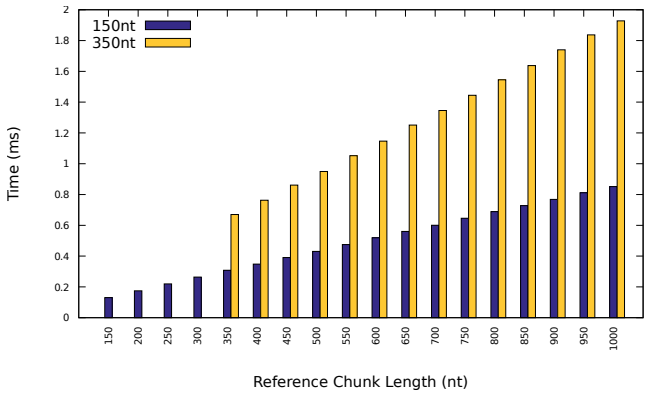

Fig. 7. Alignment score refinement time with different read lengths (150 and 350 nucleotides) and growing reference lengths, including the decryption and encryption times. Tested on the SGX enclave of a single i7-6700.

The second step in MaskAl is the score refinement of the candidate positions, computed on step 1, in enclaves. The score refinement step itself is divided into several parts: encryption and decryption of the reads, decryption of the references, reducing the decrypted reference chunk into the correct size, performing Smith-Waterman (SW) algorithm and finally encrypting and sending the result back to the requesting bio-center. As SW's time complexity is cubic, we have to reduce the workload for it to a minimum. The masked alignment with BWA has a precision of at least 98% considering a tolerable offset of +/- 50nt (see Figure 3). Therefore, we reduce the SW table size to $readLength \times (readLength + 100)$ instead of $readLength * referenceLength$. The computation time for the SW algorithm increases linearly when either the reference chunk, or the read length, increase. As Figure 7) shows, the computation time increases linearly with the length of the chunk of reference genome used during the computation. A longer read length (i.e., 350 nucleotides instead of 150) also increases the running time of the algorithm.

To complete the computation time evaluation, network communications have to be taken into account. The data MaskAl has to transfer is reduced to 4 different types (see Figure 1). As the four data flows in our architecture are using mostly unencrypted plain text, MaskAl has a very small network traffic footprint whose size is depending on the read length and amount of reads (see Figure 10).

E. Alignment time - repartition

Figure 8 shows the time distribution of MaskAl phases, considering a network bandwidth of 1Gbit/s and 96 dedicated SGX enclaves inside an enclave cloud.

Figure 9 compares the computational time for the alignment task using BWA, MaskAl and Balaur. The results show that BWA is the fastest algorithm. Our approach, MaskAl, is on average 58% slower than BWA. However, BWA runs on plaintext inputs in public clouds, and is therefore not privacy preserving. On the other hand, MaskAl is on average 87% faster than the current most competitive privacy preserving alignment algorithm, Balaur.

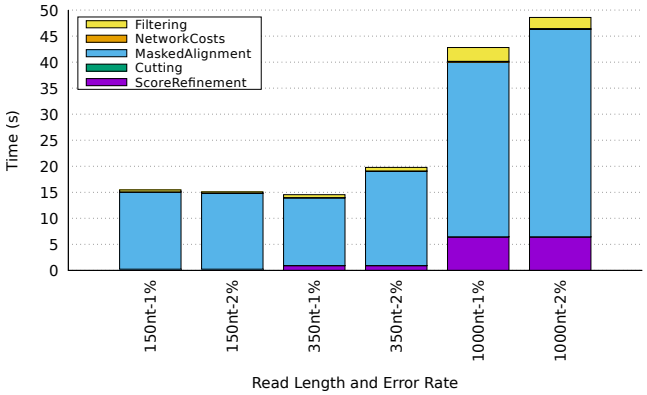

Fig. 8. Computation times of MaskAl.

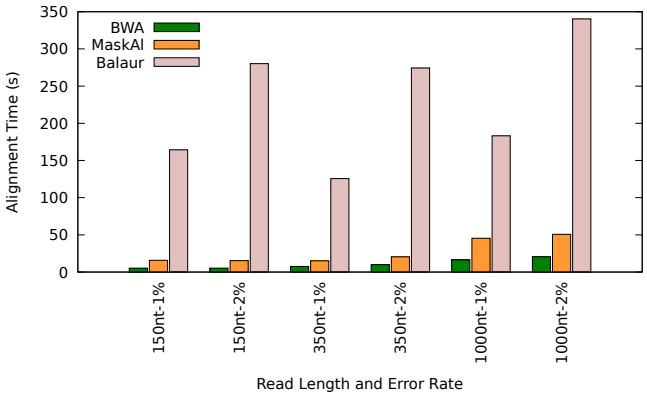

Fig. 9. Computation times of BWA, Balaur, and MaskAl. Assuming an 1GBit/s networking infrastructure for BWA, Balaur and MaskAl. Further, MaskAl is assumed to use 96 dedicated SGX-Cores.

F. Network communication

As stated in the computation time evaluation, the networking footprint for MaskAl is very small compared to Balaur. This is a result of using established plain-text alignment algorithms (BWA) with masked reads. One positive side effect of masked reads is the fact that a masked read is smaller than its unmasked pendant. Therefore, the transmission costs for the masked alignment are smaller compared to the plain text alignment of unmasked reads.

MaskAl has two important network transfers (see the overview in Figure 1): (i) the bio-center starts the masked alignment by sending the files containing the masked reads to be aligned to a public cloud. After aligning the masked reads, the bio-center receives SAM-files containing the candidate positions of the masked reads from the public cloud instances; and (ii) the bio-center sends the encrypted unmasked reads and their positions to the enclave cloud to trigger the score refinement. Finally, the bio-center receives the results of the score refinement encrypted from the enclave-cloud instances.

In contrast, Balaur uses keyed-hashes of reads. Balaur needs significantly more networking resources than MaskAl, as the hashes have a constant length to be secure. This results in the amount of data transferred staying more or less constant even for different read lengths. But nevertheless, it uses between 5.7GB and 15 GB more bandwidth than MaskAl (see Figure 10). Interestingly, Balaur uses at least the double of the bandwidth if the error rate increases from 1% to 2%, e.g., for reads of 150nt with a 2% error rate Balaur requires 16GB of transferred data, while it only transfers 7.5GB for 1% error rate.

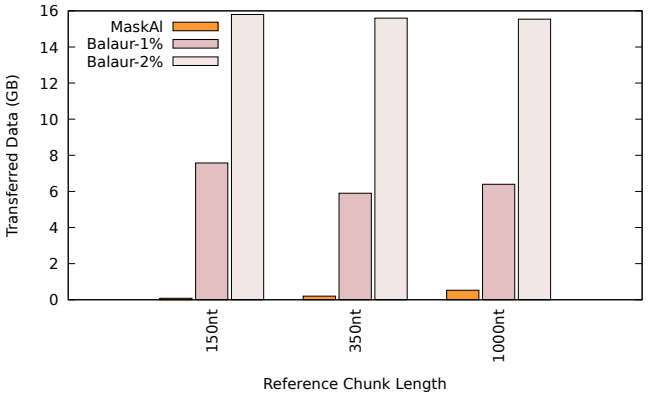

Fig. 10. Network communication costs per 100000 reads using MaskAl, BWA and Balaur.

VIII. Related Work

In contrast to the well researched area of non-privacy-preserving read mapping algorithms [42], [43], [31], [14], [32], [44], [13], the field of privacy-preserving read mapping algorithms is comparatively young. Such algorithms can be grouped into three different approaches: (i) novel cryptographic primitives; (ii) homomorphic encryption [45]; and (iii) separating the alignment process in sensitive and insensitive parts. One example for using new cryptographic primitives to assure privacy for read mapping is an approach proposed by Huang et al. [15], derived from Yao's solution with garbled circuits for his Millionaires' problem [46]. The concept of garbled circuits is proven secure [47] - however it is very slow: The Smith-Waterman algorithm implementation by Huang et al. [15] takes 415 seconds to compute two sequences each with a length of 60nt on two Intel E8400 while producing 1.17GB of network traffic.

Besides cryptographic primitives like garbled circuits, several approaches rely on homomorphic encryption. Homomorphic encryption defines for each of the common set of arithmetic and conditional operations an equivalent set of operations on the cipher text with the same effect as the plain text operations after decryption. Therefore, with homomorphic encryption it is possible to process the cipher text like plain text. Applied to the problem of sequence alignment, after encrypting the sequences as well as the reference genome, it is possible to process the encrypted data like plaintext sequence and reference.

However, the big drawback of homomorphic encryption is its high computation overhead. In addition to the high computational costs, the initial computation (encryption) has to be done on secured, private hardware (e.g. private clouds) to assure confidentiality and integrity during the encryption process. As shown in different papers [18], [48], fully homomorphic encryption is not competitive in speed when compared with traditional approaches like BLAST [32] or bowtie2 [14].

To avoid the drawbacks of homomorphic encryption and garbled circuits — namely the computation overhead and the costs to process the computation on private hardware — we propose a different approach to align reads. Our alignment's paradigm consists in differentiating the alignment process depending on whether the data handled is sensitive or not. To assure confidentiality and integrity, sensitive data is processed in private clouds. This can be done by encrypting the sensitive raw data into insensitive keyed hashes on private clouds, transferring the hashes to public clouds, and execute there the heavy read mapping operations with hashes instead of the raw genome plain text (see Chen et al. [17]). A faster implementation of a similar concept is introduced by Popic and Batzoglou in their approach, Balaur [18].

Even though Chen et al.'s [17] and Popic et al.'s [18] hybrid approaches are leaner than homomorphic or garbled circuits encryption, they suffer from several disadvantages. Besides the vulnerability of hash-based attack vectors, their main disadvantages are the need of a self-hosted, maintained and operated private cloud to assure confidentiality and integrity regarding the transition process from raw genomic data to keyed hashes. Compared to those approaches, MaskAl explores a different paradigm by relying on a filtering method [21] to identify potentially sensitive nucleotides, and to remove them from the main alignment step. After that, MaskAl executes an alignment refinement in remote SGX enclaves.

IX. Conclusion

In this paper, we presented MaskAl, a novel read alignment approach based on the partitioning of reads according to the detected sensitivity of their nucleotides. MaskAl relies as much as possible on public clouds, which may host enclaves, and only execute a low cost filtering step in a private cloud. Therefore, MaskAl improves over the state of the art of privacy-preserving alignment methods both in terms of computation overhead and network communications, even though it relies on a TEE implementation of Smith-Waterman in SGX enclaves to refine the masked alignment scores. In addition, MaskAl's accuracy is nearly as good as those of unmasked alignment methods. Since SGX restricts the secure memory per CPU to 128MB only, we limited the maximum read length to 1000nt. Nevertheless, we do not see this limitation as a problem since Intel announced the removal of this hardware restriction in future versions. Otherwise, the score refinement step can also be moved from SGX to another TEE solution. MaskAl is on average 87% faster than the most recent privacy-preserving alignment algorithm (Balaur), mostly thanks to its reduced communication overhead. In addition, MaskAl consumes significantly less RAM during data processing (in average 95% less). To conclude, MaskAl allows privacy-preserving read alignment to scale, using public clouds, to maintain a high throughput to meet the future processing requirements of the quickly emerging area of DNA analysis.

ACKNOWLEDGEMENTS

This work was supported by the Fonds National de la Recherche Luxembourg (FNR) through PEARL grant FNR/P14/8149128. We would like to thank Francisco M. Couto and Marcus Völp for fruitful discussions and comments during the course of this research.

REFERENCES

[1] K. Offit, "Personalized medicine: new genomics, old lessons," *Human Genetics*, vol. 130, no. 1, pp. 3–14, 2011.

[2] K. Kidd, A. Pakstis, W. Speed *et al.*, "Developing a snp panel for forensic identification of individuals," *Forensic science international*, vol. 164, no. 1, pp. 20–32, 2006.

[3] S. E. Levy and R. M. Myers, "Advancements in next-generation sequencing," *Annual Review of Genomics and Human Genetics*, vol. 17, no. 1, pp. 95–115, 2016.

[4] E. Ayday, J. L. Raisaro, J.-P. Hubaux *et al.*, "Protecting and evaluating genomic privacy in medical tests and personalized medicine," in *WPES*, 2013.

[5] M. Gymrek, A. L. McGuire, D. Golan *et al.*, "Identifying personal genomes by surname inference," *Science*, vol. 339, no. 6117, pp. 321–324, 2013.

[6] D. R. Nyholt, C.-E. Yu, and P. M. Visscher, "On jim watson's apoe status: genetic information is hard to hide," *European Journal of Human Genetics*, vol. 17, pp. 147–149, 2009.

[7] R. Klitzman, P. S. Appelbaum, and W. Chung, "Should life insurers have access to genetic test results?" *JAMA*, vol. 312, no. 18, pp. 1855–1856, 2014.

[8] T. . G. P. Consortium, "A global reference for human genetic variation," *Nature*, vol. 526, pp. 68–74, 2015.

[9] J. C. Venter, M. D. Adams, E. W. Myers *et al.*, "The sequence of the human genome," *Science*, vol. 291, no. 5507, pp. 1304–1351, 2001.

[10] F. L. Rocha and M. Correia, "Lucy in the sky without diamonds: Stealing confidential data in the cloud," *DSN-W*, 2011.

[11] A. Michalas, N. Paladi, and C. Gehrmann, "Security aspects of e-health systems migration to the cloud," in *IEEE Healthcom*, 2014.

[12] B. Fabian, T. Ermakova, and P. Junghanns, "Collaborative and secure sharing of healthcare data in multi-clouds," *Information Systems*, vol. 48, pp. 132–150, 2015.

[13] H. Li and R. Durbin, "Fast and accurate short read alignment with burrows–wheeler transform," *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, 2009.

[14] B. Langmead and S. L. Salzberg, "Fast gapped-read alignment with bowtie 2," *Nature Methods*, vol. 9, no. 4, pp. 357–359, 2012.

[15] Y. Huang, D. Evans, J. Katz *et al.*, "Faster secure two-party computation using garbled circuits." in *USENIX Security*, 2011.

[16] E. De Cristofaro, S. Faber, and G. Tsudik, "Secure genomic testing with size-and position-hiding private substring matching," in *WPES*, 2013.

[17] Y. Chen, B. Peng, X. Wang *et al.*, "Large-scale privacy-preserving mapping of human genomic sequences on hybrid clouds." in *NDSS*, 2012.

[18] V. Popic and S. Batzoglou, "A hybrid cloud read aligner based on minhash and kmer voting that preserves privacy," *Nature Communications*, vol. 8, 2017.

[19] S. B. Mokhtar, A. Boutet, P. Felber *et al.*, "X-search: revisiting private web search using intel sgx," in *Middleware*, 2017.

[20] R. Pires, D. Gavril, P. Felber *et al.*, "A lightweight mapreduce framework for secure processing with sgx," in *CCGrid*, 2017.

[21] J. Decouchant, M. Fernandes, M. Völp *et al.*, "Accurate filtering of privacy-sensitive information in raw genomic data," *Journal of biomedical informatics*, vol. 82, pp. 1–12, 2018.

[22] Y. Tong, J. Sun, S. S. Chow *et al.*, "Cloud-assisted mobile-access of health data with privacy and auditability," *IEEE Journal of biomedical and health Informatics*, vol. 18, no. 2, pp. 419–429, 2014.

[23] B. Yüksel, A. Küpçü, and Ö. Özkasap, "Research issues for privacy and security of electronic health services," *Future Generation Computer Systems*, vol. 68, pp. 1–13, 2017.

[24] B. Malin, *Compromising privacy with trail re-identification: the REIDIT algorithms*. Carnegie Mellon University. Center for Automated Learning and Discovery, 2002.

[25] B. Malin and L. Sweeney, "How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems," *Journal of biomedical informatics*, vol. 37, no. 3, pp. 179–192, 2004.

[26] M. Schwarz, S. Weiser, D. Gruss *et al.*, "Malware guard extension: Using SGX to conceal cache attacks," *CoRR*, vol. abs/1702.08719, 2017.

[27] F. Brasser, U. Müller, A. Dmitrienko *et al.*, "Software grand exposure: SGX cache attacks are practical," in *WOOT*, 2017.

[28] J. Götzfried, M. Eckert, S. Schinzel *et al.*, "Cache attacks on intel sgx," in *ACM EuroSec*, 2017.

[29] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[30] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet mathematics*, vol. 1, no. 4, pp. 485–509, 2004.

[31] H. Li, J. Ruan, and R. Durbin, "Mapping short dna sequencing reads and calling variants using mapping quality scores," *Genome research*, vol. 18, no. 11, pp. 1851–1858, 2008.

[32] S. F. Altschul, W. Gish, W. Miller *et al.*, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, no. 3, pp. 403 – 410, 1990.

[33] S. M. Kiełbasa, R. Wan, K. Sato *et al.*, "Adaptive seeds tame genomic sequence comparison," *Genome research*, vol. 21, no. 3, pp. 487–493, 2011.

[34] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[35] O. Gotoh, "An improved algorithm for matching biological sequences," *Journal of molecular biology*, vol. 162, no. 3, pp. 705–708, 1982.

[36] "1000 genomes project," http://www.internationalgenome.org/.

[37] "Tandem repeats database," https://tandem.bu.edu/cgi-bin/trdb/trdb.exe.

[38] H. Li, "wgsim-read simulator for next generation sequencing," 2011.

[39] M. J. Atallah, F. Kerschbaum, and W. Du, "Secure and private sequence comparisons," in *ACM WPES*, 2003.

[40] J. H. Cheon, M. Kim, and K. Lauter, "Homomorphic computation of edit distance," in *Financial Cryptography and Data Security*, 2015, pp. 194–212.

[41] F. Saeed, A. Perez-Rathke, J. Gwarnicki *et al.*, "High performance multiple sequence alignment system for pyrosequencing reads from multiple reference genomes," *Journal of parallel and distributed computing*, vol. 72, no. 1, pp. 83–89, 2012.

[42] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[43] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443 – 453, 1970.

[44] R. Li, Y. Li, K. Kristiansen *et al.*, "Soap: short oligonucleotide alignment program," *Bioinformatics*, vol. 24, no. 5, pp. 713–714, 2008.

[45] J. Baron, K. El Defrawy, K. Minkovich *et al.*, "5pm: Secure pattern matching," in *SCN*, 2012, pp. 222–240.

[46] A. C. Yao, "Protocols for secure computations (extended abstract) 1982," *University of California, Berkeley, CA*, p. 160.

[47] Y. Lindell and B. Pinkas, "A proof of security of yao's protocol for two-party computation," *Journal of cryptology*, vol. 22, no. 2, pp. 161–188, 2009.

[48] B. Wang, W. Song, W. Lou *et al.*, "Privacy-preserving pattern matching over encrypted genetic data in cloud computing," in *IEEE INFOCOM*, 2017.