Enclave-Based Privacy-Preserving Alignment of Raw Genomic Information [*]

Information Leakage and Countermeasures

Marcus Völp, Jérémie Decouchant, Christoph Lambert, Maria Fernandes, and Paulo Esteves-Verissimo
CritiX Group — Interdisciplinary Center for Security, Reliability and Trust (SnT)
University of Luxembourg, L-2721 Luxembourg
<name>.<surname>@uni.lu

ABSTRACT

Recent breakthroughs in genomic sequencing led to an enormous increase of DNA sampling rates, which in turn favored the use of clouds to efficiently process huge amounts of genomic data. However, while allowing possible achievements in personalized medicine and related areas, cloud-based processing of genomic information also entails significant privacy risks, asking for increased protection. In this paper, we focus on the first, but also most data-intensive, processing step of the genomics information processing pipeline: the alignment of raw genomic data samples (called reads) to a synthetic human reference genome. Even though privacy-preserving alignment solutions (e.g., based on homomorphic encryption) have been proposed, their slow performance encourages alternatives based on trusted execution environments, such as Intel SGX, to speed up secure alignment. Such alternatives have to deal with data structures whose size by far exceeds secure enclave memory, requiring the alignment code to reach out into untrusted memory. We highlight how sensitive genomic information can be leaked when those enclave-external alignment data structures are accessed, and suggest countermeasures to prevent privacy breaches. The overhead of these countermeasures indicate that the competitiveness of a privacy-preserving enclave-based alignment has yet to be precisely evaluated.

CCS Concepts

•Security and privacy → Hardware security implementation;

Keywords

information-flow, SGX-enclaves, DNA, alignment, privacy

[*]**Author Preprint** This work is in part supported by the Fonds National de la Recherche Luxembourg (FNR) through PEARL grant FNR/P14/8149128.

SysTEX'17, October 28, 2017, Shanghai, China
© 2017 ACM. ISBN 978-1-4503-5097-6/17/10. . . $15.00
DOI: https://doi.org/10.1145/3152701.3152707

1. INTRODUCTION

Intel's Software Guard Extensions (SGX) [12] is the latest effort to establish trustworthy execution environments (TEEs) in which sensitive parts of an application are protected from attacks on environmental components , or from their faults. Like Inktag [16] but unlike many previous approaches, SGX removes the management operating system (OS) from the trusted computing base (TCB) of sensitive application parts executing in provided TEEs (called *enclaves*). At the same time, SGX keeps the OS responsible for enclave construction and management.

Removing the management OS from the TCB is essential for protecting data as sensitive, and long living, as DNA. Any fault, or compromise, in this large complex OS may allow adversaries to exfiltrate sensitive parts of DNA, and thereby, compromise the anonymity of the individual, and the privacy of some of her own genomic traits, but also of her descendants, and even of whole populations [5]. However, this TCB reduction also comes at the cost of making certain enclave-internal actions observable and manipulatable by untrusted code in the management OS. Recent attacks [32] have demonstrated such exfiltration and manipulation possibilities. Several proposals have been formulated to mitigate their effects. For example, Gullamudi and Chong [14] suggest a language-based approach to extract sensitive application parts, forthcoming SGX version 2 will conceal sub-page granular address information, and Völp et al. [31] propose hardware extensions to reduce the number of observable states.

However, while these solutions help mitigate information leakage through objects residing inside an enclave, they remain agnostic to confidentiality breaches when data is accessed outside this protected environment. Unfortunately, during a DNA alignment enclave-external data accesses are inevitable, since the size of the reference genome, or of the data structures that encode it, exceeds by far the amount of secure memory that enclaves can access in a reasonably fast manner.

In this paper, we explain how existing alignment algorithms may leak sensitive information from reads through data structures access patterns. We look at cache side-channel attacks to refine the page-granular access patterns exposed by SGX-2, and demonstrate how clever manipulations of the reference genome encoding data structures may allow adversaries to exfiltrate sensitive parts of the reads. Finally, we detail how existing countermeasures can be used to prevent privacy leaks during reads alignment.

2. BACKGROUND AND RELATED WORK

2.1 DNA Sequencing and Alignment

Next generation sequencing machines extract genomic information from prepared samples, by multiplying the DNA, splitting these copies into smaller strands, and sampling these strands to digitize the information. The latter typically involves attaching and (e.g., optically) reading out the reaction of the base at the edge of the strand with special indicator bases, repeating the process after indicator and base have been chopped off. The result is a set of overlapping strings (called *reads*) of letter/number pairs which encode the bases Adenine, Thymine, Cytosine and Guanine (abbreviated A, T, C, G) plus a confidence in their detection. The length of the strand thereby defines the size of the read, which in modern machines ranges from a few hundred basepairs (bp) to currently up to 40000 bp [7]. The whole genome of a human amounts to 3.2 Giga-basepairs and requires less than a day to sample.

Reconstructing a genome from sampled reads can be done in two ways. First, a de-novo assembly compares sampled reads to find where they overlap, and combines them to reconstruct the genome. Second, as humans share around 97% of their DNA, one can compute the best matching position of reads in a reference genome. In this work, we shall focus on reference-based alignment.

The differences between an individual's genome and the artificial reference genome are called *variants*. It is these variants that carry sensitive information. They may reveal who we are, indicate our susceptibility to illnesses, drugs or medicine, or expose our family relationships. Indeed, Pakstis et al. showed that as few as 92 well-chosen SNPs are enough to identify anyone in the world with high probability [25]. In this work, we aim at preventing variant leakage to prevent adversaries from reidentifying individuals, or their characteristics.

Variants can be characterized as insertions, deletions or modifications of letters from the reference genome. Such differences between a genome and the reference genome motivates the use of edit distances to determine the quality of alignment matches. A typical genome [11] differs from the reference at 4.1 – 5 million locations of which the vast majority of $> 99.9\%$ are single nucleotide polymorphisms (SNPs), i.e., modifications of a single character. Of the remaining variations between 2100 and 2500 are structural variations (SVs), which break up into 1000 large deletions, 1100 are short insertions of various kinds, 160 are copy-number variations (i.e., variations in the number of copies of known sequences) and only 10 are inversions, flipping characters. The rest are short insertions and deletions (indels) of 1–10 Kbases.

Genomic alignment algorithms can be vaguely categorized into two classes:

Search-based aligners traverse the reference from promising candidate locations while computing the edit distance of the read to the reference, keeping the first/first-k/all locations whose distance is below a given threshold;

Index aligners compute a search index (e.g., a suffix tree) from the reference to more quickly identify possible locations of a read, trading higher memory consumption against better alignment performance.

Seeding as a preprocessing step may help obtain promising candidate sets for search-based aligners such as Blast [1], and also some index-based aligners. During this step, the algorithm checks whether a read contains an identifying substring (the seed), whose location in the reference is already known and stored in a location map. Seeds can be small segments of the read hashed into a location map but also more complicated patterns such as $(A\_C\_G\_T)^6ACGT$, with gaps [1] destined to tolerate variants and sequencing errors.

In this work, we investigate applied candidates from each of these classes: last [19], a successor of Blast (search-based), and bowtie [20] (index-based). More precisely, we investigate the alignment of encrypted reads in TEEs to guarantee that only trusted code obtains access to this privacy sensitive information. At the same time, the reference genome and reference-related index structures remain unencrypted and unsigned outside the TEE, because they contain no sensitive information and are too large to fit into the TEE memory.

2.2 Intel SGX Enclaves

The attacks and countermeasures we present in this paper are applicable to all TEEs where TEE-internal accesses to external, but possibly also internal data structures are observable (e.g., through page faults, cache timing channels or similar means). As an example, we focus here on Intel SGX enclaves, which we briefly summarize in the following.

Enclaves are implemented as virtual-memory regions (the ELRANGE) in the hosting application's address space. As such, they are subject to the same page-table based virtual-to-physical address translation as their host. However, rather than mapping to arbitrary frames, SGX triggers an exception if addresses in the ELRANGE map to frames outside a dedicated physical-memory area called enclave page cache (EPC), if the frame does not belong to the enclave or if a linear address is mapped to a frame that has not been accepted by the enclave (either implicitly at enclave creation time or later by the enclave executing EACCEPT). The processor also faults if the enclave fetches code from non-EPC frames.

Options for implementing the EPC include on-chip scratchpad memories, but also normal RAM, in which case a memory encryption unit ensures data integrity, confidentiality and freshness, by causing enclave-exiting faults on integrity and freshness violations. Enclaves may access data mapped to linear addresses outside the ELRANGE, however, SGX ensures that the reverse is not possible and that EPC backing store accesses do not reveal confidential information.

The processor secures control transfers into and out of enclave. However, through page-faults and controlled cache side-channel attacks [32], adversaries may learn about the addresses of memory accesses to internal and external enclave data structures. In this paper, we deviate from SGX by assuming countermeasures are in place to prevent side channels from leaking accesses targeting data structures that are located entirely in EPC memory. Still, most alignment approaches are susceptible to leakage by observing enclave-external access patterns.

2.3 Related Work

Naturally, there is a large body of works on TEE-based data processing (e.g., [24]). We relate in the following only to works on genomic-data processing. Modern encryption

[1]The example —NEAR seeding— is given here in regular-expression notation with _ denoting allowed gaps.

techniques such as homomorphic encryption [6, 13, 3] and similar approaches [4, 17, 18] can be applied to privacy-preserving alignment by directly operating on encrypted genomic information. Short of breaking the cipher, data confidentiality is preserved, however, at the cost of sacrificing performance below what is required to keep up with modern sequencing technology.

Chen et al. [9] suggest hashing short substrings and mapping them to a similarly hashed reference. Popic et al. [26] extend this approach to locality-sensitive hashing.

Cogo et al. [10] introduce a bloom filter to classify reads as sensitive or insensitive to process the former in a private cloud. Unfortunately, classifying the whole read no longer works when reads grow in size, because all but a few long reads contain sensitive variations.

Patent CA2852916A1 [29] suggests offloading genomic information processing into TEEs. However, they do not address the resource limitations of contemporary TEEs.

SAFETY [27] is a hybrid homomorphic encryption approach for later stages in the genomic information processing pipeline. It computes aggregates through a Paillier homomorphic cryptosystem while offloading more complex statistical tests into enclaves. Also focusing on a later stage, Brasser et al. [8] discuss a cache-side channel attack on PRIMEX [21], a hash-based search index tool for locating primers in polymerase chain reactions.

3. PRIVACY-PRESERVING ALIGNMENT

We assume enclaves are deployed in the usual authenticated boot, sealed memory and communication infrastructure [2] capable of authenticating the enclave configuration in the cloud, and of equipping it with the key material it needs to securely communicate with the read-transmitting sequencing system. We assume enclaves decrypt the received reads, align them, and return the alignment results in encrypted form. Short of exploiting vulnerabilities in the cipher, its implementation or in SGX, adversaries can therefore only learn about sensitive reads by observing the execution of an enclave and its access patterns.

3.1 Detailed Definition of the Problem

We are concerned with adversaries aiming at exfiltrating genomic variations from the access patterns that alignment algorithms generate in TEEs. Such information potentially allow the adversaries to learn about the identity of a donor, and infer other privacy sensitive information in the donor's DNA. The adversaries try to obtain these information from reads used in the trusted alignment algorithm. Reads are transmitted encrypted and only ever decrypted by the enclave, which is supposed to keep reads confidential. However, data structures used during alignment contain only public information, and some of them are too large to fit in the constrained enclave memory. In particular, the reference genome, and the data structures derived from it, are public information. We assume these data structures are not encrypted, otherwise, the overall performance of enclave-based read alignment would probably not be competitive. Seed patterns are also publicly known, but whether or not a specific pattern matches a read constitutes sensitive information. In particular, a seed may only match a read because the latter contains a sensitive variation.

Most variations, their locations and sizes are also known and can be obtained, for example, from the 1000 Genomes project [30] which provides the variations of individuals. A further source of variations is the Short Tandem Repeats (STR) Database [22], which lists known patterns and locations where short 3-5 base-pair sequences are repeated several times. Knowing the number of repetitions of certain STRs allows reidentifying individuals and must therefore be considered privacy sensitive. We are confident that most variations have been discovered because the number of newly found variations in the 1000 genomes project decreased from 21 million in the initial set of 2010 to 100.000 in the 2016 set of Phase 3. Whether or not a variation is sensitive depends on the information it encodes. However, variations located in the vicinity of encoding variations may reveal this information, as shown for example by Nyholt et al. [23] who deduced the variant of Dr. Watson's removed Alzheimer gene. For the purpose of this paper, we say that variations are *sensitive* if they allow for privacy impacting attacks (e.g., re-identification), leaving the development of more refined definitions as future work.

The tasks at hand are: (1) to understand how adversaries may deduce which variations are present in a genome; and (2) to construct countermeasures to prevent adversaries from succeeding in these attacks. These tasks are complicated by the fact that because of the way DNA is sequenced, a lot of overlapping reads are generated, some of which may include errors. Alignment, having to reconstruct the genome despite read errors, must therefore consider all reads to mask these errors whereas adversaries need only be able to analyze a subset to gain sufficient confidence in the presence of a certain variation.

In this paper, we do not consider privacy attacks due to later disclosure of the aligned data (e.g., in genomic studies).

3.2 Search-based Alignment

Search-based tools such as Blast [1], and Last [19], compute the local alignment of a read relative to the reference by applying one of the variants of the Smith-Waterman-Gotoh algorithm [28, 15]. Given the length of the human genome, this algorithm is only applied in combination with search-space pruning heuristics, such as seeding. We return to seeding in Section 3.4. Here, we first analyze leakage of a naive implementation, which traverses through the entire reference genome. The results apply to pruned locations as long as the sizes of the constructed data-structures exceed available enclave memory.

To align a read to a reference, Smith-Waterman-Gotoh constructs a so called *DP matrix* with one row for each letter of the read and one column for each (considered) location of the reference genome. Cells store alignment scores, which are computed iteratively as a combination of the three cells in the top left corner ($H_{i,j}$) and as a function of the read and reference letter $S(a_i, b_j)$ of the cell's row/column pair. Figure 1 illustrates this algorithm. More precisely, the alignment score of cell $H_{i,j}$ is equal to $max(H_{i-1,j-1} + S(a_i, b_j), H_{i,j-1} + w_{insert}, H_{i-1,j} + w_{delete}, 0)$, where w_k are penalties for inserting a letter in, or respectively deleting a letter from, the reference, and $S(a_i, b_j)$ rewards or penalizes the alignment depending on whether or not the reference letter a_i matches the read letter b_j. Together with the score, Smith-Waterman-Gotoh stores the cell from which the high score originated which.

After the matrix has been populated, threshold exceeding cells indicate possible end locations of optimal local align-

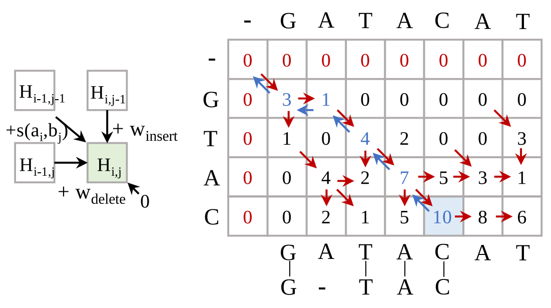


Figure 1: Smith-Waterman-Gotoh algorithm. The score of cell $H_{i,j}$ is computed as the maximum of 0 and the top left preceding cells weighted with a match/mismatch score $S(a_i, b_j)$, an insertion score w_{insert}, and a deletion score w_{delete}. Shown here are linearly weighted indels.

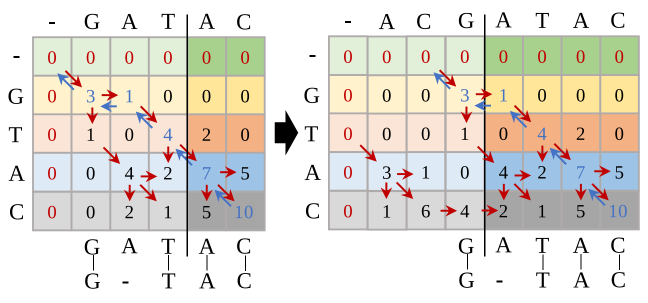


Figure 2: Inserting characters in the reference shifts the A-deletion in $G - TAC$ to the next cacheline. Shown are the cache-colored matrices for the original and the modified reference genome. After inserting two additional characters, the cells (G, G) and (G, A) span different cachelines, which pinpoints the deletion.

ments, while the backtracking paths from these locations reveals the shape of the alignment (in terms of matches, indels or character replacements).

Attack Vectors: Clearly since the reference genome is already too large, the size of the DP matrix exceeds enclave memory, even if only a part of the DP matrix is populated. Exposing the DP matrix values in clear, indels become visible by following the alignment path. From the size of indels, known variations with differing sizes can be distinguished. For example, a one letter insertion at a location where one or two letter variants are known to appear reveals can be identified. The same information can be obtained up to the granularity of the observable block accesses (e.g., a cacheline) by observing the backtracking pattern. Matches, sequencing errors or mismatches (e.g., due to SNPs) create a diagonal traversal pattern. More precisely, if we assume a row-wise stored DP matrix and a block size of $bc = 4$ cells per block[2], diagonal backtracking advances through exactly bc rows before changing to the next bc columns. For insertions, backtracking traverses vertically, which means more rows are accessed before switching columns and deletions cause horizontal traversals and an earlier column change. The precision at which indels are revealed is bc. However, adversaries may improve the precision of deletions by artificially inflating the reference genome when replaying the alignment request until the deletion point moves across cacheline boundaries. Figure 2 illustrates this attack.

Adversaries can replay alignments either by resubmitting intercepted client requests to the server, or by relying on overlapping reads which align to nearby locations. Whereas constructing the DP matrix leads to a regular access pattern. The location of alignment is revealed by the access pattern created during backtracking.

The attacks discussed so far only reveal the location of alignment and the nature of variations whose size is unique and different from the variation present in the reference genome. However, manipulation of the reference genome allows adversaries to also identify the letters of variations of the same size by enumerating all possible variants in an artificial reference. Given a read length r and two fixed-size variations at location l, the artificial reference needs to copy the $2r$ letters around l and the letters of the two variations to two different locations in the artificial reference. Observing a backtracking pattern in either one of the two locations reveals the individual's variation. In case the alignment scores in both locations are above the threshold, non-variation letters must be changed in replays until only one of the two locations remains above the threshold.

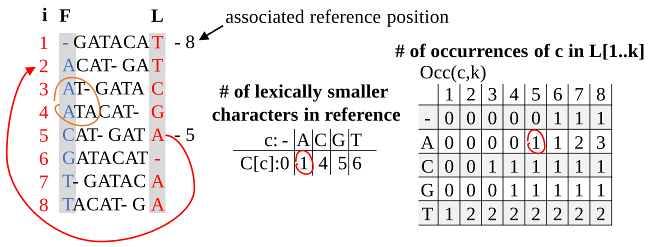
Smith-Waterman-Gotoh in Enclaves: One obvious countermeasure, if we assume non-observability of enclave-internal accesses, is to relocate the DP matrix entirely into enclave memory. Checking alignments immediately once cell scores exceed the threshold, and further considering only short indels, it is possible to reduce the size of the DP matrix by recycling the columns located beyond the deletion range. Assuming further a high number of matches, it is possible to further shrink the matrix, keeping only cells near the diagonal. Such reduced matrices start to fit in enclave memory, but continue to exceed today's fast on-chip memory capacities. For example, with 40Kbp reads and limiting insertions and deletions to 10Kbp each, the diagonalized DP matrix shrinks to 2 x 10Kbp diagonals with max 40Kbp + 10Kbp cells each (i.e., 100MBytes storage).

Attack Vectors: Without having access to the cell information, adversaries have to infer backtracking locations from cache access patterns. If we further assume such patterns are concealed in an augmented SGX implementation, the only attack vectors that remain are based on the observable location in the reference genome while constructing the associated fraction of the DP matrix, and the time that the alignment procedure spends at this location. In case the current location contains a threshold exceeding cell, backtracking becomes necessary and the enclave takes more time until it advances in the reference. Unfortunately, the traditional countermeasures against this attack (i.e., clock obfuscation and timing-leak transformations) are ineffective or too costly. For example, to equalize the time spent over a fragment of the reference genome, backtracking needs to be applied to all locations (irrespective of their score). Enumerating all variants in an artificial reference therefore remains a privacy threat.

3.3 Index-based Alignment

Index-based aligners, such as bowtie [20], preprocess the

[2] Extraction methods for column-wise or diagonally stored DP matrices can be obtained in a similar fashion.



c:	-	A	C	G	T
C[c]:	0	1	4	5	6

	1	2	3	4	5	6	7	8
-	0	0	0	0	0	1	1	1
A	0	0	0	0	1	1	2	3
C	0	0	1	1	1	1	1	1
G	0	0	0	1	1	1	1	1
T	1	2	2	2	2	2	2	2

Figure 3: FM index of the reference of Figure 1. The index is comprised of the Burrows Wheeler Transformation of the reference (vector $L[i]$) and of two tables $C[c]$ and $Occ(c, k)$ counting the number of occurrences of lexically smaller characters respectively of the character c in $L[1 \ldots k]$. The index allows reconstructing the first column F from L as $F[LF[i]] = L[i]$ with $LF[i] = C[L[i]] + Occ(L[i], i)$.

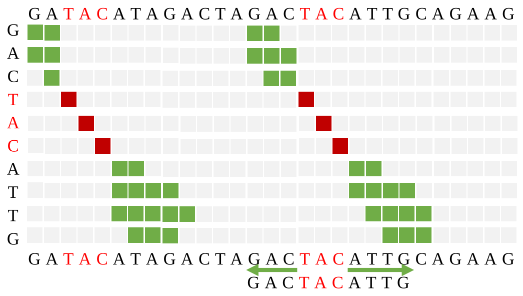

G A T A C A T A G A C T A G A C T A C A T T G C A G A A G
G
A
C
T
A
C
A
T
T
G
G A T A C A T A G A C T A G A C T A C A T T G C A G A A G
G A C T A C A T T G

Figure 4: Smith-Waterman-Gotoh to extend seed locations.

reference genome to construct index data structures for faster suffix search. Figure 3 shows bowtie's FM index applied on the example presented in Figure 1. Alignment indexes are based on the Burrows Wheeler Transformation (BWT) of the reference, which is the last column $L[i]$ obtained from lexicographically sorting all rotations of the reference string. For example, rotating $-GAT \ldots AT$ by one position, we obtain $T - GAT \ldots A$, which after lexicographic sorting becomes the 7^{th} string. The index stores $L[i]$ the two lookup tables $C[c]$ and $Occ(c, k)$ and for some indexes i of $L[i]$ the location of this character in the reference. $C[c]$ stores the number of lexicographically smaller characters that exist in the reference. When lexicographically sorted, this is exactly one less than the index of the first string starting with c (i.e., $F[C[c] + 1]$ is the first index occurrence of c in the Burrows Wheeler Transformation. $Occ(c, k)$ denotes for each index k how often c occurs in the vector L before k.

The actual alignment proceeds by searching all occurrences of suffixes, which are sufficiently similar to the read. Starting from the end of the read with the last character c and the interval $[start \ldots end] = [C[c] + 1 \ldots C[c + 1]]$ of indexes in L (where $c+1$ is the character following c in lexicographical order), the interval is successively adjusted to $[C[c] + Occ(c, start - 1) + 1 \ldots C[c] + Occ(c, end)]$ For example, the interval for suffix 'AT' is obtained by starting with $[C['T'] + 1 \ldots C[T + 1]] = [7 \ldots 8]$ and refining it to $[C['A'] + Occ('A', 6) + 1 \ldots C['A'] + Occ('A', 8)] = [3 \ldots 4]$. Because of the lexicographical sorting of the rotated reference, suffixes correspond to dense intervals in L. Associated positions are then used to translate positions in L back to the reference by traversing L until a location with associated reference position is found. For example, mapping last to first column character of the prefix AT at location 3 with $LF[i] = C[L[i]] + Occ(L[i], i)$, we obtain the index of the previous character at $L[i]$ whose location in the reference genome is stored. Hence, the location of CAT is 5 and AT is found at 6.

Attack Vectors: Because the FM index is only constructed from the reference, $L[i]$, $C[c]$ and $Occ(c, k)$ contain only public information. With only four bases plus the empty character '-' and column wise storage of $Occ(c, k)$, no direct information about the nature of looked up bases can be obtained from $C[c]$ or $Occ(c, k)$. The first easily fits a cacheline whereas the latter can be stored columnwise and padded such that columns do not cross cacheline boundaries. Therefore, accessing $Occ(c, k)$ only reveals k but not c.

More problematic is the direct revelation of read characters when searching for suffix locations. Dense intervals imply that subsequent indexes of $L[i]$ stand for the same character $F[i]$. It is therefore likely that adversaries will learn about $F[i]$ by just learning about the cacheline granular access. Since the FM index only encodes the reference, variations cause some backtracking when the suffix contains a variation letter. The accessed location $L[i]$ prior to backtracking reveals this letter. For example, looking up the suffix TA in ATA reveals the variation letter T through location 8 although the match would interleave 5 and 2 with CA and ACA before matching $TACA$ at 8.

3.4 Seeds

Seeds are short sequences from the reference sequence, which are carefully chosen for their matching properties and searched in reads. They are used to accelerate both search and index-based aligners by allowing them to focus on promising locations, from which they extend the match to the read. Figure 4 shows this extension scheme for Smith-Waterman-Gotoh based search alignment. From a seed (red diagonal), the DP matrix is extended in both directions until the computed scores drop significantly.

Attack Vectors: Seeding inherits the attack vectors of search-based algorithms for the extension, and of index-based algorithms if like an FM-index is used as location map for the candidate set. A seed may match because a read contains a specific variation. Such sensitive seeds should then not be used.

3.5 Countermeasures

Several countermeasures are imaginable to mitigate the identified attack vectors. Cogo et al. [10] suggest circumventing the problem by not aligning privacy sensitive data in the cloud. However, relying on the filter they suggest to classify reads as sensitive and insensitive would work only to up to 30 bases reads, since the majority of larger reads (produced by modern machines) contain sensitive parts.

Traditional side channel mitigation strategies (e.g., equalizing the cache access pattern, possibly in combination with preemption control [31], or memory randomization [8]) are difficult to apply to TEE-external data structures or lead to significant performance impacts, in particular when applied to large data structures. We gave an example of such an equalizing method when discussing Smith-Waterman-Gotoh backtracking, which would need to be applied to all cells irrespective of whether the alignment score exceeds the matching threshold.

The nature of the majority of variants could only be revealed through reference modification. Effective countermeasures to a large class of attacks therefore include mechanisms to ensure the integrity of the reference genome and of derived indexes. Unfortunately, these mechanisms (e.g., keyed hashes) bear non-negligible costs and require copying in part of the data structure to TEE memory to avoid TOCTOU attacks. Encrypting a secret shuffle of the reference or of the FM index vector $L[i]$ randomizes locations, however, at significant costs.

4. CONCLUSIONS AND FUTURE WORK

This paper investigated leakage of sensitive DNA variations during TEE-based alignment of raw genomic information. We identified several attack vectors based on observing accesses to TEE external data, through which adversaries may exfiltrate this privacy-violating information, even if accesses to TEE-internal data structures are concealed.

Directions for future work include exploring substring classifiers for raw genomic information and the data privacy one can obtain from aligning such classified reads. We are also interested in privacy-preserving solutions for later stages of the genomic information processing pipeline.

References

[1] S. F. Altschul et al. “Basic local alignment search tool”. In: *Journal of molecular biology* 215.3 (1990), pp. 403–410.

[2] I. Anati, S. Gueron, S. P. Johnson, and V. R. Scarlata. *Innovative Technology for CPU Based Attestation and Sealing*. Tech. rep. Intel Corp., Aug. 2013.

[3] M. J. Atallah, F. Kerschbaum, and W. Du. “Secure and private sequence comparisons”. In: *WPES*. 2003.

[4] M. J. Atallah and J. Li. “Secure outsourcing of sequence comparisons”. In: *International Journal of Information Security* 4.4 (2005), pp. 277–287.

[5] M. Backes et al. “Simulating the Large-Scale Erosion of Genomic Privacy Over Time”. In: *GenoPri*. 2016.

[6] J. Baron et al. “5pm: Secure pattern matching”. In: *Security and Cryptography for Networks*. Springer, 2012, pp. 222–240.

[7] P. Biosciences. *Pacific Biosciences Introduces New Chemistry with Longer Read Length*. Oct. 2013.

[8] F. Brasser et al. “Software Grand Exposure: SGX Cache Attacks Are Practical”. In: *WOOT*. 2017. URL: https://www.usenix.org/conference/woot17/workshop-program/presentation/brasser.

[9] Y. Chen, B. Peng, X. Wang, and H. Tang. “Large-Scale Privacy-Preserving Mapping of Human Genomic Sequences on Hybrid Clouds.” In: *NDSS*. 2012.

[10] V. V. Cogo, A. Bessani, F. M. Couto, and P. Verissimo. “A high-throughput method to detect privacy-sensitive human genomic data”. In: *WPES*. 2015.

[11] T. 1. G. P. Consortium. “A global reference for human genetic variation”. In: *Nature* 526.7571 (Oct. 2015), pp. 68–74.

[12] V. Costan and S. Devadas. *Intel SGX Explained*. MIT.

[13] E. De Cristofaro, S. Faber, and G. Tsudik. “Secure genomic testing with size-and position-hiding private substring matching”. In: *WPES*. 2013.

[14] A. Gollamudi and S. Chong. “Automatic Enforcement of Expressive Security Policies using Enclaves”. In: *OOPSLA*. 2016.

[15] O. Gotoh. “An improved algorithm for matching biological sequences”. In: *Journal of molecular biology* 162.3 (1982), pp. 705–708.

[16] O. S. Hofmann et al. “InkTag: Secure Applications on an Untrusted Operating System”. In: *ASPLOS*. 2013.

[17] Y. Huang, D. Evans, J. Katz, and L. Malka. “Faster Secure Two-Party Computation Using Garbled Circuits.” In: *USENIX Security*. 2011.

[18] S. Jha, L. Kruger, and V. Shmatikov. “Towards practical privacy for genomic computation”. In: *IEEE Security and Privacy*. 2008.

[19] S. Kiebasa et al. “Adaptive seeds tame genomic sequence comparison”. In: *Genome Research* 21.3 (2011), pp. 487–493.

[20] B. Langmead, C. Trapnell, M. Pop, S. L. Salzberg, et al. “Ultrafast and memory-efficient alignment of short DNA sequences to the human genome”. In: *Genome biol* 10.3 (2009), R25.

[21] M. Lexa and G. Valle. “PRIMEX: Rapid identification of oligonucleotide matches in whole genomes”. In: *Bioinformatics* 19.18 (2003).

[22] NIST. *STRBase: Short Tandem Repeat DNA Internet DataBase*. http://www.cstl.nist.gov/biotech/strbase/.

[23] D. R. Nyholt, C.-E. Yu, and P. M. Visscher. “On Jim Watson’s APoE status: genetic information is hard to hide”. In: *Eur. J. Hum. Genet.* 17 (2009), pp. 147–149.

[24] O. Ohrimenko et al. “Oblivious Multi-Party Machine Learning on Trusted Processors”. In: *USENIX Security*. 2016.

[25] A. J. Pakstis, W. C. Speed, R. Fang, and F. C. e. a. Hyland. “SNPs for a universal individual identification panel”. In: *Human genetics* 127.3 (2010), pp. 315–324.

[26] V. Popic and S. Batzoglou. “A hybrid cloud read aligner based on MinHash and kmer voting that preserves privacy.” In: *Nature comm.* 8 (2017), p. 15311.

[27] N. Sadat et al. *SAFETY: Secure gwAs in Federated Environment Through a hybrid solution with Intel SGX and homomorphic encryption*. Mar. 2017.

[28] T. F. Smith and M. S. Waterman. “Identification of common molecular subsequences”. In: *Journal of molecular biology* 147.1 (1981), pp. 195–197.

[29] *Systems and methods for protecting and governing genomic and other information*. Patent CA2852916A1.

[30] The 1000 Genomes Project Consortium. “An integrated map of genetic variation from 1,092 human genomes”. In: *Nature* 491 (2012), p. 1.

[31] M. Völp et al. “Avoiding Leakage and Synchronization Attacks Through Enclave-Side Preemption Control”. In: *SysTEX*. Trento, Italy, 2016. ISBN: 978-1-4503-4670-2.

[32] Y. Xu, W. Cui, and M. Peinado. “Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems”. In: *IEEE Security and Privacy*. 2015.