Large-Scale Privacy-Preserving Mapping of Human Genomic Sequences on Hybrid Clouds

Yangyi Chen, Bo Peng, XiaoFeng Wang, Haixu Tang
School of Informatics and Computing
Indiana University Bloomington, IN, USA
{yangchen, pengbo, xw7, hatang}@indiana.edu

Abstract

An operation preceding most human DNA analyses is read mapping, which aligns millions of short sequences (called reads) to a reference genome. This step involves an enormous amount of computation (evaluating edit distances for millions upon billions of sequence pairs) and thus needs to be outsourced to low-cost commercial clouds. This asks for scalable techniques to protect sensitive DNA information, a demand that cannot be met by any existing techniques (e.g., homomorphic encryption, secure multi-party computation). In this paper, we report a new step towards secure and scalable read mapping on the hybrid cloud, which includes both the public commercial cloud and the private cloud within an organization. Inspired by the famous "seed-and-extend" method, our approach strategically splits a mapping task: the public cloud seeks exact matches between the keyed hash values of short read substrings (called seeds) and those of reference sequences to roughly position reads on the genome; the private cloud extends the seeds from these positions to find right alignments. Our novel seed-combination technique further moves most workload of this task to the public cloud. The new approach is found to work effectively against known inference attacks, and also easily scale to millions of reads.

1 Introduction

The rapid advance of human genomics technologies has not only revolutionized life science but also profoundly impacted the development of computing technologies. At the core of this scientific revolution is the emergence of the high-throughput Next Generation Sequencing (NGS) technologies: today, a single sequencer can generate millions of short DNA sequences (called *reads*) with 35 to 250 base-pairs (bp) long over one billion nucleotides [40]. To interpret these sequences, the most important step is to align them to a public human DNA sequence (called *reference genome*) to identify their genetic positions and other features, e.g., whether they belong to human or human microbes. This operation, known as *read mapping*, is a *prerequisite* for most DNA sequence analyses [54] (e.g., SNP discovery, genotyping, personal genomics, etc. [49]). It is also a step involving intensive computation, given the huge size of the reference genome - 6 billion nucleotides, and the complexity of the mapping operation - calculating *edit distances* between reads and all the substrings on the reference genome. With the fast-growing sequence data produced by NGS, the demands for mapping such data are increasingly hard to be met by the computing power within organizations [52]. A trend widely believed to be inevitable is outsourcing this computation to low-cost commercial clouds [52]. For example, Amazon Elastic Compute Cloud (EC2) provides different types of computing instances at a price as low as 0.015 dollar per hour [2]. This development, however, brings in privacy risks: on the one hand, NGS reads can be used to identify sequence donors [39], a serious privacy threat that can lead to denying their access to health/life/disability insurance and educational/employment opportunities; on the other hand, today's cloud providers do not offer high security assurance and tend to avoid any liability [3]. To protect donors and avoid legal troubles, the NIH so far disallows any datasets involving human DNA to be handed over to the public cloud.

Secure computation outsourcing. Addressing this urgent demand requires the techniques capable of sustaining large-scale secure computation outsourcing. Unfortunately, none of the existing approaches are designed for the computation of such an enormous scale. Particularly, homomorphic encryption, secret sharing and secure multi-party computation (SMC) are far too heavyweight to handle read mapping. For example, a privacy-preserving protocol [17] takes about 5 minutes to calculate the edit distance between two 25-element sequences through homomorphic encryption and oblivious transfers. Such a task can be computed much

more efficiently by SMC techniques, thanks to the recent progress in this area [33, 35]. However, even the state-of-the-art SMC implementation needs about 4 seconds to process two 100-element sequences [33]. Also, secret-sharing based approaches [17, 18] all incur data exchanges between share holders when aligning a single sequence pair. Given that a mapping task routinely evaluates the edit distances for 10^{15} sequence pairs, all these approaches are simply not up to this job. Alternatively, one may consider anonymizing the sequence data by aggregating the reads from multiple individuals or adding noise. These approaches are known to be vulnerable to statistical re-identification attacks [32, 34, 46, 56], a threat that prompted the NIH to remove all aggregated DNA data [12] from the public domain.

Special features of the problem. We feel that existing generic approaches all fail to appreciate the special features of the problem, which can be leveraged to build a practical solution. Actually, the edit distance considered in read mapping is small, typically no more than 6 for standard 100-bp reads [29]. This is because human genetic variation is believed to be small, below 0.5%. In practice, the differences between a read and its counterpart on the reference genome mainly come from sequencing errors, whose rate is typically 2-3%. Therefore, a privacy-preserving technique that works on small edit distances (caused by the variations and the errors) should be enough for handling most sequence-analysis tasks.

The cloud also has distinctive features. It is extremely good at performing simple operations over a large amount of data, as evidenced by the pervasiveness of *MapReduce*-based computing services [23]. Also interesting is the way that commercial clouds are used in practice: they often serve as a receiving end of the computation "spill-over" from an organization's internal system when its computing resources are about to deplete. This way of computing, which involves both the private cloud within an organization and the public commercial cloud, is called *hybrid cloud computing* [26]. The hybrid cloud has already been adopted by most organizational cloud users and is still undergoing rapid development. It also presents a new opportunity that makes practical, secure outsourcing of computation tasks to untrusted environments possible: for example, we can split a task, delegating to a commercial cloud a large amount of relatively simple computation over encrypted data, like exact string matching, while letting the user's private cloud work on a small amount of relatively complicated computation such as calculating edit distances.

Our approach. Given that read mapping is one of the most important and pervasive operations in DNA sequence analysis [54], we believe that a special solution is justified here to enable secure and practical outsourcing of this computation to the low-cost commercial cloud. In this paper, we show how the aforementioned features can be used to serve this purpose. Our idea is based upon the observation that the edit distance of two sequences can actually be evaluated by checking the exact matches between some of their substrings, an operation easy to secure on the public cloud, when all we care about is whether the distance is below a small threshold. This is described by the well-known *seed-and-extend* strategy [19], which first matches a *seed*, a segment of a read with one-$(d+1)$th of the read's length given an edit distance d, to the substrings on the reference genome[1], and then *extends* from these substrings to align the whole read. Our approach splits the mapping task along these two stages, delegating them to public and private clouds respectively: the public cloud searches for the exact matches between the keyed hash values of seeds and those of substrings (called *l-mers* for a substring of length l) on the reference genome to determine the possible positions of reads, while the private cloud extends from these positions to find the right alignments of these reads (which involves calculating edit distances under a threshold [28]).

However, this simple approach cannot guarantee to offload most of the mapping workload to the public cloud. This is because the traditional seed-and-extend approach used in computational genomics [38] has always considered the situation when seeding and extension happen on the same system: there has never been a need to move the workload from one stage to the other. Particularly, when seeds are too short, due to a relatively large edit distance (e.g., 6 for 100-bp reads), they randomly match a large number of genetic locations on the reference genome. As a result, a lot of extensions have to be done on the private cloud. In our research, we come up with a novel solution, which transforms the computation workload of the private cloud to the spatial cost for the public cloud. More specifically, our approach uses *seed combinations* to ensure that only a relatively small number of extensions happen to each read.

We conducted a security analysis of our technique over the reference genome, particularly on the threat of frequency analysis. This risk is found to be minimum by our experimental analysis over the whole genome, due to the special structure of human genome: most of its l-mers become unique when l grows over 20. Our performance evaluation on a cross-the-country cloud testbed with real human microbiome data indicates that the technique is sufficiently practical for real-world use: it mapped 10 million reads to the whole reference genome in a few hours, outsourced over 97% of the workload to the public cloud and maintained an acceptable level of overall computational, communication and spatial overheads.

Contributions. Here we summarize the contributions of the paper:

- *Practical privacy-preserving mapping technique*. We

[1] At least one seed will match a substring on the region within d edits from the read.

propose a new technique that makes an important step towards secure and scalable DNA sequence mapping. Running on the hybrid-cloud platform, our approach works effectively against inference attacks and has the capacity to process millions of reads with most of the workload being outsourced to the public cloud. This opens the great potential to use the cheap computing resources of the commercial cloud to meet the urgent demand of analyzing a large amount of NGS data. Although the technique was designed for read mapping, lessons drawn from this study can have broader implications: the idea like combining lightweight cryptography/aggregation and transforming complicated computation to easy-to-protect but data intensive computation for the cloud could find other applications in the domains that need practical privacy protection.

• *Implementation and evaluation*. We implemented our technique over *Hadoop*, and evaluated the privacy and the performance of our prototype using real human microbiome data, on a large-scale, across-the-country cloud testbed.

The focus of our research is confidentiality, as it is the main hurdle to outsourcing read-mapping computation to the public cloud and therefore urgently needs practical solutions. Although integrity of cloud computing is no doubt important, so far people are willing to live with the risk that their tasks may not be handled correctly by the cloud. This is evidenced by the fact that mapping of non-human reads already happened on EC2 [47], while the tasks involving human data are not allowed to be given to the public cloud.

Roadmap. The rest of this paper is organized as follows. Section 2 describes security challenges in outsouring anonymized read data; Section 3 presents our secure mapping techniques and analyzes the privacy protection they offer; Section 4 reports an evaluation on the performance of our approach; Section 5 surveys related prior research and Section 6 concludes the paper.

2 Backgrounds

2.1 Read Mapping

The DNA data produced by a next generation DNA sequencer consists of millions of reads, each typically including 100-120 nucleotides. These reads are randomly sampled from a human genome. To interpret them, their genetic locations must be found, which is achieved through read mapping. Specifically, given a set of reads, a reference genome (i.e., a long DNA sequence) and a threshold, the mapping operation aims at aligning each read to a substring on the reference genome such that the edit distance between the read and the substring does not exceed a threshold. This operation positions each read to its genetic location, which is necessary for most human DNA analyses, including SNP discovery, genotyping, gene expression profiling (e.g. RNA-seq), comparative genomics, personal genomics and others [49]. It is also a critical step for analyzing the DNA data of human microbes [8], serving to sanitize such data by removing the DNA contamination from human hosts.

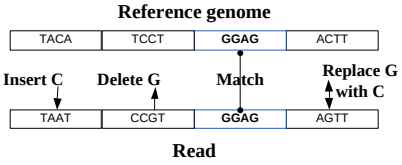

Figure 1. An Example (two sequences with edit distance 3)

Seed-and-extend. As discussed before, a prominent feature of read mapping is that the edit distance is small between a read and the reference substring it should be aligned to: since the genetic variation between different humans is typically below 0.5% and sequencing errors are about 2-3%, almost all the mapping tasks look at a distance threshold no more than 6 for standard 100-bp reads [29], which is sufficient for tolerating both the variations and the errors. For such a small edit distance, alignments can be efficiently found through *seed-and-extend* [19], a method that has been extensively used in high-performance mapping systems like BLAST [15], BLAT [36], Bowtie [37], RMAP [50, 51] and CloudBurst [47]. The method is based upon the observation that for a sequencing read partitioned into $d + 1$ segments (seeds) of length l, if it has at most d errors, then at least one of its $d + 1$ segments must match a substring at the genetic location (on the reference genome) the read should be mapped onto [19]. Therefore, one can use the segment to roughly locate the read on the reference (the seeding stage), and then extend from these possible locations to determine where it indeed belongs (the extension stage). Figure 1 gives an example: one of the 4 seeds of a 16-bp read matches a 4-mer on a reference substring when the edit distance between them is 3; once this match is found, we can align these two strings to check whether the distance is indeed no more than 3. Our idea is to let the public cloud take care of the seeding, roughly locating reads, which enables the private cloud to quickly extend each read at a relatively small set of positions (rather than at each of the 6 billion positions on the reference genome). The extension is done by a linear algorithm that calculates the edit distances no more than a threshold [28].

2.2 Privacy threat

Threat. The main threat to human genomic data is identification of the individual the DNA comes from. Since

such data is often produced by clinic studies, its donor, once identified, could be linked to the disease under the study, which can have serious consequences such as denial of access to health/life insurance, education, and employment. The Health Insurance Portability and Accountability Act (HIPAA) requires removal of explicit identifiers (such as name, social security number, etc.) before health data can be released [11]. This protection, however, was found to be insufficient for genomic data, as re-identification can still happen through examining the genetic markers related to the donor's observable features (i.e., phenotypes) after the genetic locations of reads are recovered, a concern shared by the NIH [12].

The genetic variation most widely-used for such identification is single-nucleotide polymorphism (SNP). SNP occurs when a single nucleotide (`A`, `T`, `C` or `G`) differs between the members of a species. It can take one of two *alleles*, either 0 (major) or 1 (minor). This variation has been employed by all existing re-identification techniques [32, 34, 46, 56]. Theoretically, other variations, including rare alleles [48] and copy number variation (variations over sections of DNA) [13], could also be used to identify an individual. However, a critical barrier for the identification based on rare variations is that they are not mapped at the whole genome scale in any reference population. Also note that to pose the same level of the threat, those variations need a larger population: given that rare-allele frequencies are much lower than those of SNPs, a population 100 or even 1000 times larger than the HapMap population (about 270 people) may need to publish their whole genomes for accurate estimate of the frequencies, which may not be likely given the increasing awareness of the privacy implication of DNA data. Note that under the protection of our technique, an adversary cannot infer the rare alleles from the data on the public cloud (i.e. the keyed-hash values of l-mers, not their content), because without known rare allele frequencies in a reference population, the hash values of these l-mers are indistinguishable from those not belonging to human, such as those of microbes. For copy-number variations, a recent study [22] shows that their identification power is way lower than SNPs, due to their small numbers (only about 5,000), much lower density in comparison to SNPs, and continuous distributions [42]. Therefore, it is commonly believed that the threat can be easily mitigated by simple approaches like data aggregation. *So far, no known re-identification techniques use such variations*.

Challenges in outsourcing read mapping. To enable read mapping to happen on the public cloud, we need to either encrypt the data or anonymize it, making it unidentifiable. Unfortunately, only a few cryptographic approaches [17, 18, 20, 33, 35, 55] support secure calculation of edit distances, and all of them are too expensive to sustain a large-scale computation like read mapping (Section 5). Less clear is the efficacy of simple data anonymization techniques such as aggregation, noise adding and data partition. These techniques have long been used to protect genomic data. As a prominent example, Genome-Wide Association Studies (GWAS) typically use DNA microarrays to profile a set of pre-determined SNPs from a group of patients (called *case*) to study the genetic traits of their common disease. The SNPs from different case individuals were often aggregated into a mixture, from which nothing but the counts (or equivalently, the frequencies) of different SNP values (alleles) can be observed and used for statistical analysis. Such aggregate data was deemed safe to release. However, recent studies show that the data is actually vulnerable to a type of re-identification attacks [32, 34, 46, 56]: given the allele frequencies of a *reference* population, which can be acquired from public sources such as the International HapMap Project [9], and a DNA sample from an individual, her presence in the case population can be determined from the aggregate data through a statistical test.

Different from the microarray data studied in the prior work [32,34,46,56], the SNP sets covered by the reads from two persons often differ significantly, due to the randomness in sequencing. In our research, we systematically evaluated the effects of these anonymization techniques on read data using a near-optimal test statistic proposed in the prior research [32]. Our findings show that such data is equally vulnerable to the re-identification attack. For example, to ensure that no more than 10% of a case group can be identified at a confidence of 0.99, we found that the reads from about 38000 individuals, with 1 million reads each, need to be aggregated. The details of this study is presented in Appendix 7.1.

3 Secure Read Mapping

3.1 Overview

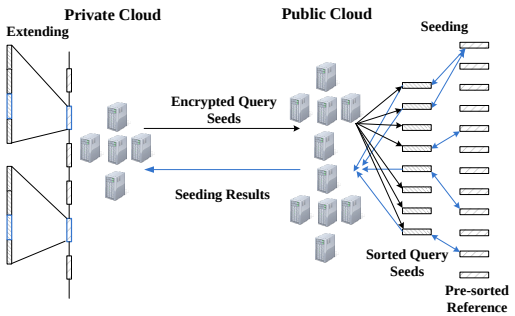

Figure 2. High-level design of our system.

Our design. The high-level design of our techniques is illustrated in Figure 2. Our approach is built upon a hybrid

cloud: the public commercial cloud is delegated the computation over the keyed hash values of read data, while the private cloud directly works on the data. Our idea is to let the private cloud undertake a small amount of the workload to reduce the complexity of the secure computation that needs to be performed on the public cloud, while still having the public cloud shoulder the major portion of a mapping task. To this end, we divided the task according to the seed-and-extend strategy [19]. The seeding part roughly locates a read on the reference genome by matching a small segment of it to the substrings on the genome. For example, given an edit distance of 3 and reads of 100 bps, we can partition each read into 4 segments (seeds), each of 25 bps. At least one of these seeds will match a 25-mer on the read's counterpart on the reference genome. Searching for this position is done over the keyed-hash values of the seeds and 25-mers: we first extract all 25-mers from the reference genome, remove repeated sequences and fingerprint the *distinctive* ones with a cryptographic hash function and a secret key; these reference hash values are compared with the hash values of the seeds on the public cloud; all the matches found are reported to the private cloud, which extends the reads from the genetic positions (on the reference genome) indicated by these matches to find an optimal alignment for each read, using a threshold edit-distance algorithm (Section 3.2).

This basic approach works when the seeds are long enough to suppress random matches: for example, most 25-mers are unique across the genome, so a 25-bp seed often locates a read at a small set of positions, which reduces the workload of calculating edit distances between millions of reads and billions of reference 25-mers to the extensions that just need to happen at millions of positions. However, the seeds can be short in practice. For example, given an edit distance of 6, the seed for an 100-bp read has only 14 bps and often matches thousands of positions. To address this problem, we improve the basic approach to perform the seeding over keyed-hash values for 2-combinations of 12-bp seeds, which significantly reduces the workload of the private cloud at a spatial cost easily affordable by modern clouds (Section 3.3).

The privacy assurance of our techniques is evaluated by the amount of information the public cloud can infer from the data it observes. To achieve an ultra-fast mapping, we adopted keyed hash, which allow the public cloud to unilaterally determine whether a match happens. This, however, brings in the concern about a frequency analysis that infers the content of l-mers by counting the matches their hashes receive. To assess this risk, we conducted a whole genome study, which demonstrates that the adversary actually cannot achieve any re-identification power through such an analysis (Section 3.4).

Adversary model. We consider an adversary who aims at re-identifying the individuals related to read data. As discussed before, we focus on this re-identification threat because it is the main privacy concern for releasing protected health information [32, 34, 46, 56], which includes sequencing reads, and therefore the major barrier to moving read mapping to the public cloud. We assume that the private cloud is trustworthy while the nodes on the public cloud can be compromised and controlled by the adversary. Also, we assume that the adversary has a DNA sample of a *testee*, the person she wants to identify from a read dataset, and a reference population genetically similar to those whose reads are inside the dataset. Access to such knowledge is widely considered to be a very strong assumption [30] that gives the adversary advantages. It serves as the standard background information in all studies on re-identification threats to genomic data [32,34,46,56], and the foundation for evaluating whether such data can be safely released [10, 46]. Finally, note that all we care about here is to prevent the adversary from identifying read donors, rather than to ensure the success of the computation, which may not be achieved when the adversary controls the public cloud.

3.2 Computation Split

In this section, we elaborate the basic design of our technique, which is also summarized in Table 1.

Data preprocessing. To perform the seeding on the public cloud, we need to compute the keyed-hash values for both the reference genome and individual seeds. Specifically, given a keyed hash function $H_K()$ with a secret key K, our approach first fingerprints all *distinctive* l-mers α_i on the reference genome: $H_K(\alpha_1)$, $H_K(\alpha_2)$, $\cdots$ and then sends their hash values (in a random order) to the public cloud. We remove all repeats here to mitigate the risk of a frequency analysis. Depending on the length l, l-mers have different levels of repetitions on a human genome: for example, we found that more than 80% of 24-mers are unique on the reference genome. Note that such data processing and transfer only need to be done once to enable the public cloud to map a large number of read datasets using the hashed reference. For each dataset, we compute keyed hashes for the seeds s_j extracted from each read, $H_K(s_1)$, $H_K(s_2)$, $\cdots$, $H_K(s_{d+1})$, randomly permutate this list and then deliver it to the public cloud for the seeding operation. Our implementation adopts SHA-1 and a 256-bit secret key, and only uses the first 10 bytes of an l-mer's hash as its fingerprint for the comparison. The rest bytes are XORed with the information for locating the l-mer on the reference genome.

Computing on the public cloud. The seeding task delegated to the public cloud is as simple as comparing all the hashes of the seeds with those of the reference l-mers and reporting the indices of the matched sequence pairs (i, j) to the private cloud. The only problem here is the scale

Table 1. Privacy-Preserving Read Mapping: the Basic Approach

- *Generating keyed hashes for reference l-mers (one-time cost).* Given an edit-distance threshold d and a read length λ, set $l = \frac{\lambda}{d+1}$. For each distinctive l-mer α_j on the reference genome, compute $H_K(\alpha_j)$ (j is unrelated to the l-mer's position). Remove all the repeats and send the distinctive hash values to the public cloud.
- *Generating keyed hashes for seeds.* On the private cloud, for each read in a given read dataset, break it into $d+1$ seeds with a length of l each. Compute $H_K(s_i)$ for every seed s_i and send all these hash values to the public cloud.
- *Seeding.* On the public cloud, compare the hashed seeds to the hashed references. For all $H_K(\alpha_j) = H_K(s_i)$, send (i, j) to the private cloud.
- *Extension.* On the private cloud, for every matched pair (i, j), extend the read including s_i from the genetic location pinpointed by α_j on the reference genome to check whether the edit distance (between the read and the substring on that location) is no more than d.

of this computation, which involves millions upon billions of string comparisons. One way to achieve a fast seeding is to build an index for the reference genome, as typically done by the fast-mapping software designed to work on standalone systems. This approach, however, needs a huge amount of memory and cannot be easily parallelized. Our implementation uses ultra-fast sorting, which the cloud is good at, to do the seeding. Specifically, the public cloud pre-sorts the reference l-mers according to their hash values. For every batch of seed hashes, the cloud first sorts them and then merges them with the sorted l-mer hashes to find matches. This strategy has also been adopted by CloudBurst [47], a famous cloud-based (yet insecure) mapping system. Today's cloud can already support high-performance sorting: for example, Terasort [44] running on Hadoop attained a sorting speed of 0.578 terabytes per minute.

Computing on the private cloud. The private cloud extends the seeds at the locations where matches happen. These locations are recovered from the indices of seed hashes and the l-mer hashes they match, as reported by the public cloud. For this purpose, two look-up tables are needed: one maps the hash of an l-mer to its occurrences on the reference genome, and the other maps that of a seed to its read. The first table is relatively large, at least 10 GB. We reduced the size of the table using the features of human genomes. When l goes above 20, most l-mers are unique across the reference genome. Particularly, only a small portion (below 20%) of 24, 25-mers repeat. For every unique l-mer α_i, our approach keeps its location information directly on the last 6 bytes of $H_K(\alpha_i)$. Specifically, let θ_i be these bytes; we XOR the location of α_i, L_i, onto θ_i: $\pi_i = \theta_i \oplus (I_i||L_i)$, where I_i is a one-byte indicator for the uniqueness of the l-mer. Once this l-mer is matched by a seed (that is, the first 10 bytes of the seed's hash matching the first 10 bytes on the l-mer's hash), L_i is recovered from π_i using θ_i, which comes from the hash of the seed and is kept on the private cloud. For those still in the table, we organize them according to the indices of their hashes to ensure that only sequential access happens when searching the table.

When the read dataset is relatively small (10 million reads or less), its look-up table, which does not go above 1.2 GB, can often be accommodated in the memory. In the table, we also keep the last 6 bytes of seed hashes for decrypting the location information of the l-mers they matched. To handle a large dataset, our design encrypts the read information R_j for a seed s_j using a simple stream cipher such as AES CTR. Specifically, we first compute the key-stream $\sigma_j = E_{K'}(V||j)$, where $E()$ is an encryption algorithm, V is an initial vector and K' is another secret key, and then use the first 10 bytes of the stream to do the encryption: $\tau_j = \sigma_j \oplus (R_j||\theta_j)$, where θ_j is the last 6 bytes of $H_K(s_j)$. This ciphertext is concatenated with the first 10 bytes of the seed's hash, and given to the public cloud. Once a match is found between the hashes of α_i and s_j, it sends (π_i, τ_j, j) to the private cloud for recovering L_i and R_j.

The workload of the private cloud is determined by the number of extensions it needs to perform for each read. As discussed before, when l is no smaller than 20, most l-mers are unique and thus the reads whose seeds match them often need to be extended only a few times. Among the rest of l-mers that reoccur on the genome, some of them are actually part of longer repetitive substrings, and only need to be extended once for all these reoccurrences. This can be achieved by compressing the reference genome according to its 100-mers: we can identify all distinctive 100-mers and extend the reads on them instead of the whole genome. Also important here is an efficient extension algorithm. Our implementation utilizes a threshold dynamic programming algorithm [28] to compute the edit distance no more than a threshold d. This algorithm's complexity is only $O(d\lambda)$, where λ is the length of the read.

Discussion. This application of seed-and-extend works well when the seeds are at least 20 bps. Given standard 100-bp reads, this means that the edit distance we are look-

Table 2. Privacy-Preserving Read Mapping: Seed Combinations

- *Generating keyed hashes for combined l-mers (one-time cost).* Set $l = \frac{\lambda}{d+2}$, with a distance threshold d and a read length λ. For every λ-mer (substrings of λ long) on the reference genome, compute the keyed hash values for all the 2-combinations of the l-mers it contains: $H_K(\alpha_1||\alpha_2)$, $H_K(\alpha_1||\alpha_3)$, $\cdots$, $H_K(\alpha_{\lambda-l}||\alpha_{\lambda-l+1})$. Remove all the repeats and send the distinctive hash values to the public cloud.
- *Generating keyed hashes for seed combinations.* On the private cloud, break each read into $d+2$ equal-length seeds. Compute the keyed hashes for all 2 combinations of the $d+2$ seeds: $H_K(s_1||s_2)$, $\cdots$, $H_K(s_{d+1}||s_{d+2})$. Send the hashes for all the reads to the public cloud.
- *Seeding.* On the public cloud, compare the hashes of combined seeds to those of combined l-mers. For any two hashes that match, send their indices (i, j) (i for the hash of the combined seed and j for that of the combined l-mer) to the private cloud.
- *Extension.* On the private cloud, for every matched pair (i, j), extend the read associated with the combined seed from the genetic location identified by the combined l-mer to check whether the edit distance is no more than d.

ing at should not go above 4. Our research shows that less than 20% of 24-mers re-occur. To prepare for a mapping task, the private cloud must compute the keyed hashes for reference l-mers, which only needs to be done once, and seed hashes for every dataset. Using a high-end desktop (2.93 GHz Intel Xeon), we found that SHA-1 achieved a throughput of 100 million 25-mers per minute using a single core. In other words, a typical dataset with 10 million reads can be proceeded around a minute. Fingerprinting the whole reference genome took longer time, about 7 minutes using 8 cores. However, this only needs to be done once. Note that SHA-1 is not known for its speed. We chose it in our implementation just for simplicity. Many other cryptographic hash functions perform much better [7]. Our evaluations (Section 4) show that this approach moved the vast majority of the workload to the public cloud, compared with the situation when the whole computation is done within the private cloud.

3.3 Combined Seeds

Short seeds. When edit distance goes up to 6, the 7 14-bp seeds of a read often align it to hundreds or even thousands of possible positions for extensions. For example, in a microbiome filtering task described in Section 4, our experiment shows that on average 895 extensions need to be made before an alignment within the edit distance of 6 could be found for one read. To reduce the number of matches, our idea is to use multiple seeds: given an edit distance d, we can partition a read into $d + 2$ seeds, of which at least 2 will have exact matches on the region the read should be aligned to. For example, a 100-bp read, once partitioned into 8 12-bp seeds, can use 2 seeds (totally 24 bps long) to find out the 100-mers to which its distance may not exceed 6. Given the total length of such combined seeds, most reads can be located at a few genetic positions: in the microbiome filtering task mentioned above, the 2-combination of 12-bp seeds successfully aligns a read within 28 extensions on average. A straightforward implementation of this idea, however, forces the private cloud to intersect a large number of positions randomly matched by the short seeds for each read before the reference substrings including both seeds can be identified. This intersection operation often incurs a significant overhead, which sometimes even exceeds the cost for simply running the whole task locally. As a result, the private cloud often has to shoulder most of the mapping workload.

Mapping 2-combinations. Our answer to this challenge is a novel design built upon the special features of the cloud. Today's clouds are designed for data intensive computations, and can easily store and process terabytes of data at a low cost, as long as the operations on such data are simple and parallelizable. This property allows us to trade the spatial cost on the public cloud for the reduction in the computing workload on the private cloud, converting the intersection operation to the string matching that happens between the keyed-hash values of 2-seed combinations and those of l-mer combinations. Specifically, for every 100-mer on the reference genome, we save to the public cloud *distinctive* hashes for all the 2-combinations of its l-mers α_i: $H_K(\alpha_1||\alpha_2)$, $H_K(\alpha_1||\alpha_3)$, $\cdots$, $H_K(\alpha_2||\alpha_3)$, $\cdots$, $H_K(\alpha_{100-l}||\alpha_{101-l})$. Given a read dataset, the private cloud also fingerprints all the 2-combinations of $d+2$ seeds s_j for each read: $H_K(s_1||s_2)$, $\cdots$, $H_K(s_{d+1}||s_{d+2})$. These combined-seed hashes are compared to those of the l-mer combinations on the public cloud to help locate the reads on the reference genome. This approach is summarized in Table 2.

To perform this seeding operation, the public cloud needs to accommodate the keyed hashes for both reference l-mer combinations and combined seeds. Each 100-mer contains $101 - l$ different l-mers and totally $\frac{(101-l)(100-l)}{2}$

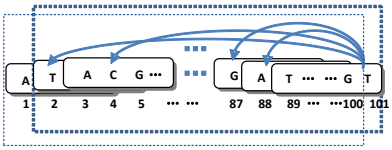

Figure 3. 12-mer Combinations in 100-bp Windows on the Genome

combinations. For example, there are 3916 combinations of 12-mers within a 100-bp window. However, the total size of the reference will not grow that much, as the seeds within two overlapping 100-mers are also heavily overlapped: from Figure 3, we can see that whenever the window right shift by one bp, only one new l-mer has been created, which brings in an additional $100 - l$ combinations. Therefore, the total size of the reference actually increases by roughly $100 - l$ times. In the above example, 12-mer combinations are about 88 times of the total size of all reference 12-mers. Using the 16 bytes of the 20-byte output of SHA-1, the keyed hashes for the reference sequences have about 6.8 TB. *Storing, transferring and processing this amount of data are extremely economic for today's commercial clouds*. For example, to keep such data at Amazon Simple Storage Service (S3), the NIH only needs to spend $890 per month [4]. Transferring the data to the EC2 is completely free [4]. In practice, a typical approach is simply to mail a hard drive, a standard service adopted by the S3 when bandwidth is low [5]. Note that this is just a one-time cost, and data storage and transfer at that scale are quite common for today's cloud users. Operations on this reference include sorting and merging with the hashes of combined seeds, which can also be efficiently done through ultra-fast sorting algorithms [44]. On the other hand, the seed combinations cause a much smaller rise in the spatial cost: the number of 2-combinations of $d+2$ seeds is $\frac{(d+1)(d+2)}{2}$, just $\frac{d+1}{2}$ times the size of the seeds. For example, the keyed hash data of 12-bp seed combinations for 10 million 100-bp reads has about 2.8 GB, roughly 10 times the size of the read dataset.

The private cloud needs to compute the hashes and deliver them to the public cloud for each mapping task. Given merely $\frac{d+1}{2}$ times of the increase in the size of the data, the overheads for computing and transferring those hashes can be easily afforded. Since the combined seeds often are sufficiently long ($\geq$ 20 bps), most of them are distinctive across the reference genome: as an example, our research shows that nearly 70% of combined 12-mers (24 bps long) are unique. This helps reduce the number of extensions performed on the private cloud. To further avoid unnecessary extensions, our approach uses a strategy that for each read, we first extend the combined seed with a unique match. This works particularly well for the task like microbiome filtering, which stops to remove a read as soon as it is found similar to a human 100-mer. A major engineering challenge on the private cloud side is the sizes of the look-up tables. The one for finding reads (from the combined seeds) is still okay, and can always be replaced with the encoding technique as described in Section 3.2. The other one, which maps l-mer combinations to their positions on the reference genome, needs to be expanded by nearly 40 folds, and has a size of roughly 400 GB. To work on this table efficiently, we can partition it into a set of sub-tables and distributed them to multiple nodes on the private cloud. As discussed in Section 3.2, the content of the original table is organized according to the index order of the hashes for l-mer combinations on the public cloud, for the purpose of sequential data access from the hard drive. Here, we further ask the public cloud to group the matches it found into several bins with regard to the index ranges of the sub-tables, and then dispatch these bins to different nodes for distributed table look-ups and extensions on the private cloud. Another approach is simply encrypting the genetic locations for the l-mer combinations that repeat less than 10 times on the reference genome, and saving them on the public cloud. Whenever the hash value of one such combination is matched, the ciphertext of its locations is sent back to the private cloud. This strategy trades bandwidth consumption for the saving of data processing time on the private cloud.

Discussion. Using combined seeds, we essentially outsource the computing burden of intersecting the matches produced by short seeds to the public cloud, which further reduces the proportion of the overall workload the private cloud needs to undertake. This has been justified by our experimental study (Section 4). Also, although computing the hash values for all 12-mer combinations takes about 11 hours and a half using 8 cores (2.93 GHz Intel Xeon), the operation only needs to be performed once. The cost for processing each dataset is still very low: for example, 280 million seeds for 10 million reads took 6 minutes to hash.

3.4 Privacy Analysis

Our study. Our approach exposes nothing but keyed hash values of seeds and l-mers to the public cloud, from which the adversary cannot directly recover reads. We further ensure that only the hashes of distinctive l-mers are disclosed, making them hard to distinguish by an observer. Under such protection, what is left to the public cloud is just the counts of the exact matches that individual reference hashes (those of l-mers or combinations) receive from the hashes of seeds or combined seeds. Therefore, the adversary's best chance is leveraging such information and the background knowledge at her disposal, i.e., the genomes of a reference population and a DNA sample from the testee, to determine the

presence of the testee's DNA in the read dataset she cannot directly access. We analyze the threat in this section.

As discussed in Section 2.2 and 3.1, we made two assumptions in our study. First, we assume the aforementioned background knowledge for the adversary. This would not be necessary if we wanted to achieve differential privacy [25], a privacy goal that does not rely on any assumption of background knowledge. However, such a privacy guarantee is often too strong to attain in practice, particularly in the case of read mapping where the genetic locations of reads and the distributions of their hashes are not known in advance and expensive to get. Also, the background information we used has been assumed in all prior studies on re-identification risks in human genomic data [32, 34, 46, 56], as it reflects the special features of the data, and serves as the foundation for evaluating when it can be released [10]. Actually, even this assumption has already been complained of as being overpessimistic [30] by the genomics community, because the knowledge it assumed often cannot easily come by in practice. Our objective is to show that even under such an assumption, which is strongly in favor of the adversary, she still cannot acquire identifiable information from the public cloud. Second, our analysis focuses on SNPs, as for the time being, other genetic variations cannot be effectively used for re-identification during read mapping (see Section 2.2).

l-mer based re-identification. The public cloud only observes the frequency with which the keyed hash value of each reference l-mer has been matched. This frequency is actually the l-mer's average rate of occurrence across genome datasets. The only thing the adversary can do is trying to map each hash value to its plaintext l-mer, according to their frequencies (i.e., the average rate of occurrence) calculated from seed-hash datasets and public genome data (e.g., the HapMap population and the reference genome) respectively. In most situations, however, this attempt will not yield unique answers, as many l-mers share same frequencies: for example, roughly 5 billion 24-mers on the reference genome are unique; many of them have same average rates of occurrence (e.g., those carrying single SNPs with identical allele frequencies) in a population. In the end, what the adversary can get is just a mapping between a set of hash values and a set of plaintext l-mers with the same average rate of occurrence, assuming that she learnt the rate from her long-term observations. Within a pair of such sets, the adversary cannot determine which hash indeed belongs to a given l-mer through the frequency analysis, as each of such hash values and l-mers has an identical frequency. Here, we use *bin* to describe this set pair, i.e., a set of l-mers grouped by the frequency they share and their corresponding set of hash values. Note that this is the best the adversary can achieve without considering the relations among the hash values. Such relations are actually very difficult to establish in practice, as elaborated in Appendix 7.3.

Consider the h bins $B_{k\in[1,h]}$ the adversary is able to identify from all the seed-hash datasets she sees. Given a specific seed-hash set (the *case* group) that involves the seeds from multiple individuals, these bins are the only attributes at the adversary's disposal for a re-identification attack. Specifically, what she can do is to find out the aggregated frequency of all the hash values within B_k for the case group. For example, suppose that only two hash values η_1 and η_2 are in B_k and they totally show up 1000 times in the 1-million-seed case group; the frequency of B_k becomes $\bar{f}_k = 0.001$ for the case. Then, the adversary figures out the aggregated frequency of the l-mers in the bin for a *reference* population (e.g., the Hapmap group), $\bar{F}_k$, and that for a DNA sample from the testee, ρ_k. In the example, $\bar{F}_k = 0.0012$ if the two l-mers in B_k, l_1 and l_2, occur totally 1200 times in 1 million l-mers of the reference population[2]. In this case, the best the adversary can do is to run the most powerful statistical test over $\bar{f}_k$, $\bar{F}_k$ and ρ_k for all h bins to determine whether the testee is present in the *case* group: i.e., whether the hash values of the testee's seeds (which the adversary cannot see) are included in the seed-hash dataset. If we treat these bins as a set of attributes like SNPs, this re-identification problem does not fundamentally differ from that studied in prior research [32, 46]. Therefore, the test statistic reported by the prior work also works here:

$$\bar{D} = \sum_{k=1}^{h} \bar{D}_k = \sum_{k} [|\rho_k - \bar{F}_k| - |\rho_k - \bar{f}_k|] \qquad (1)$$

$\bar{D}$ is called *Homer-like test*, as it is a direct application of Homer's test [32], which works on the allele frequencies of SNPs, to the re-identification over the frequencies of bins ($\bar{f}_k$, $\bar{F}_k$ and ρ_k). This test is known to be close to optimal [46] in re-identifying the testee from the case group given the aforementioned background knowledge. The truly optimal test is the famous log likelihood ratio test [46], according to the Neyman-Pearson lemma:

$$\bar{T} = \sum_{k=1}^{h} \bar{T}_k = \sum_{k} [log(Pr_k^C(\rho_k)) - log(Pr_k^R(\rho_k))] \quad (2)$$

Let $\bar{\rho}_k^C$ be the aggregated rate of occurrence for a case individual's seeds (l-mers) in B_k over all her seeds. In Equation 2, $Pr_k^C(\rho_k)$ represents the cumulative probability for seeing case members whose $\bar{\rho}_k^C$ are even less likely to observe than ρ_k (the frequency of the testee's B_k) according to the distribution of $\bar{\rho}_k^C$ in the case group, which we pessimistically assume that the adversary knows. Similarly,

[2]It is meaningless to check the frequencies of η_1, η_2, l_1 and l_2 for the case/reference/DNA sample, as the hashes cannot be linked to the l-mers.

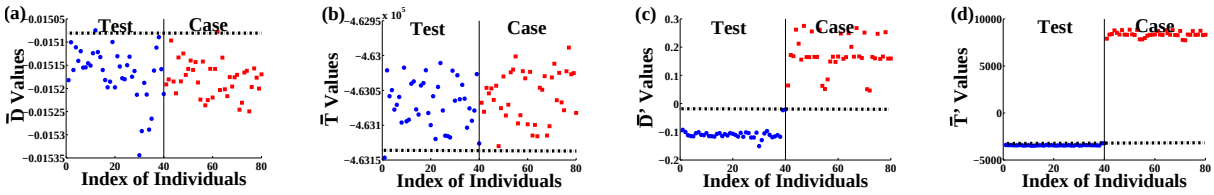

Figure 4. Power Analysis and Comparison. On bin aggregated l-mer frequencies: (a) the Homer-like statistic test $\bar{D}$ (Equation 1) (b) the likelihood ratio test $\bar{T}$ (Equation 2). On SNP allele frequencies: (c) Homer's test [32] $\bar{D}' = \sum_{k=1}^{h} \bar{D}'_k = \sum_k [|\rho'_k - \bar{F}'_k| - |\rho'_k - \bar{f}'_k|]$, where ρ'_k is the testee's allele for SNP k, and $\bar{F}'_k$ and $\bar{f}'_k$ are the major allele frequencies for the SNP on the reference and the case populations respectively. (d) the likelihood ratio test [46] $\bar{T}' = \sum_{k=1}^{h} \bar{T}'_k = \sum_k [log(\bar{P}^C_k(\rho'_k)) - log(\bar{P}^R_k(\rho'_k))]$, where $\bar{P}^R_k = \bar{F}'_k$ if ρ'_k is major allele and $\bar{P}^R_k = 1 - \bar{F}'_k$ otherwise, and $\bar{P}^C_k = \bar{f}'_k$ if ρ'_k is a major allele and $\bar{P}^C_k = 1 - \bar{f}'_k$ otherwise. All the statistics were computed for each of 40 case individuals (blue dots) and 40 test individuals (red squares). The distributions of $\bar{D}$ and $\bar{T}$ in these two groups are indistinguishable in (a) and (b), while those of $\bar{D}'$ and $\bar{T}'$ are completely separated in (c) and (d). The dash lines represent the 1% false positive rate level. The individuals identified at this rate are *above* the lines for all statistics except for $\bar{T}$, where the identified are *below* the line.

$Pr^R_k(\rho_k)$ is the cumulative probability for the reference individuals with regards to ρ_k. Note that to avoid computing complicated joint distributions, Equation 2 treats bins as being independent from each other. In practice, the correlations among the vast majority of bins are indeed weak, given the fact that each of them typically covers many SNPs and only in rare cases, a small portion of the SNPs in one bin also appear on the l-mers in other bins. Even when strongly correlated bins are found, the adversary still cannot build a stronger log likelihood ratio test (the most powerful test) over them, simply because identifying related hash values is very difficult (Appendix 7.3), not to mention determining the joint distributions over these values in different case bins, a prerequisite for the test.

Power analysis. Given the background knowledge, the two test statistics are the *most powerful* identification tools available to the adversary. Here we report a study that evaluated the identification powers of these statistics over real DNA data from the largest population available in the public domain. In this study, we compressed the whole reference genome into 372,869,332 distinctive 24-mers that involve SNPs. The selected 24-mers were further classified into 7,260,607 bins according to their frequencies. For simplicity, this was done in a way that gives advantage to the adversary. Specifically, the 24-mers involving a single SNP were grouped into bins according to their frequencies, which depended not only on their rates of occurrence across the reference genome but also on the allele frequencies (3 digits of precision) of their SNPs. For those associated with more than one SNP, we just created a unique bin for each of them, assuming that they were identified by the adversary.

The reference individuals in our study were acquired from the reference human genome. To produce realistic human genome sequences for multiple individuals, we randomly set the alleles for the SNP sites on the reference genome according to their allele frequencies reported by the HapMap. $\bar{F}_k$ in the Homer-like test (Equation 1) and the distribution in the likelihood ratio test (Equation 2) were estimated from 100 such sequences, which constituted the reference group. We used the YRI population (80 individuals) on the HapMap [9], the largest population whose DNA data is available in the public domain, to construct a case group and a *test group* (a population including neither case nor reference individuals). Each of these two groups had about 40 individuals. We sampled 10 million reads from each individual in the case group to compute $\bar{f}_k$ and the distribution of B_k frequencies. Then, we repeatedly ran $\bar{D}$ (Equation 1) and $\bar{T}$ (Equation 2) for 40 times over randomly selected case/test groups with individual group members as testees, and found that each time the case and the test individuals were completely indistinguishable. An example is presented in Figure 4 (a) and (b), which illustrates the power of $\bar{D}$ and $\bar{T}$ in one experiment. For $\bar{D}$, at 1% false positive level (denoted by the dash line), one in the case and one in the test group (i.e., a false positive) were identified (above that line), indicating equal true positive and false positive rates, so no statistical power was achieved. For $\bar{T}$, only a test individual was identified (below the line), which is a false positive.

We further compared this outcome with the identification powers the adversary can get in the absence of the new protection we propose here, i.e., when the DNA data of the case individuals was just aggregated, the standard protection the NIH took. Over the aggregated data, the adversary observes the allele frequencies of different SNPs and can therefore

run both tests over these frequencies [32,46] using the same background knowledge (the reference and the DNA sample from the testee). In our experiment, we ran these tests on the SNPs of the same populations. The results, as presented in Figure 4 (c) and (d), show that both tests easily separated the case and test populations. In contrast, these tests were completely ineffective upon the keyed hashes our approach exposes to the public cloud (Figure 4 (a) and (b)). This strongly indicates that our techniques offer effective protection against the most powerful known re-identification attacks. In Appendix 7.2, we show that $\bar{D}$ and $\bar{T}$ also cannot achieve a higher power on the combined seeds.

4 Performance Evaluation

4.1 Experiment Setting

Our evaluation was performed over a microbial filtering task [8]. The sequences extracted from human microbes include the DNA information of their hosts, which, if not taken out, will not only contaminate the outcome of a microbiome analysis but also disclose the identities of the donors the microbes come from. Therefore, one of the most important read-mapping tasks is to compare the reads from microbiome datasets to the reference genome, to identify and remove those belonging to humans. For the time being, this task is still undertaken by the NIH internal servers, but there are strong demands to move the computation to the low-cost public cloud given the privacy of the donors is protected [52].

Data. We utilized a real microbiome dataset collected from a fecal sample of a human individual [45]. The dataset contains 10 million reads, totally 250 MB, a data scale typical in today's microbe projects. In our research, we added to the dataset 500,000 human reads collected from the reference genome, a typical level of human contamination (5%) in microbiome data, as the original dataset was already sanitized, being removed of human sequences. These human reads were randomly sampled from Chromosome 1 (*Chr1*), the largest chromosome with 252.4 million bps, Chromosome 22 (*Chr22*), the smallest one with 52.3 million bps, and the whole genome with 6 billion bps respectively. They were further randomly adjusted to simulate the sequencing error and mixed with the microbiome dataset to build three test datasets (with human reads from Chr1, Chr22 and the whole genome respectively). Over these datasets, we ran our prototype under three scenarios, mapping these 10 million reads to Chr1, Chr22 or the whole genome.

Clouds. These datasets were mapped on *FutureGrid* [43], an NSF-sponsored large-scale, cross-the-country cloud testbed. The public cloud we used includes 30 nodes with 8-core 2.93 GHz Intel Xeon, 24 GB memory, 862 GB local disk and Linux 2.6.18. Our private cloud is a single node with the same software/hardware settings. We also evaluated the bandwidth use of our prototype on the 40 MBps link between the private cloud and the public cloud.

4.2 Results

In the experiments, we ran our prototype to filter the reads on the hybrid cloud, using edit distances of 3 and 6. The overheads incurred during each step of the computation was measured and compared with those of *CloudBurst*[3] [47], the most famous cloud-based mapping system, which is also a standard service of Amazon MapReduce [6]. The results are presented in Table 3 and 4.

Data preparation (one-time cost). When the distance was 3, we extracted from our "reference genome" (Chr1, Chr22 or the whole genome) distinctive 24-mers for each of the experiment scenarios, and generated keyed hash values for these references using SHA-1 with a 32-byte secret key on the private cloud. It took 29 seconds to work on Chr22, 213 seconds on Chr1 and a little more than one hour on the whole genome using a single core. For the distance of 6, we first identified 12-mers, which were further combined within 100-bp windows, as described in Section 3.3. The combined 12-mers (24 bps long in total) were then fingerprinted. This time, all 8 cores were put in use to generate the hash values. The overall time to have this job done was estimated to be 11 hours when it came to the whole genome. All these reference hashes were delivered to the public cloud, which further sorted them for preparing the mapping jobs. This sorting time varied over the scales of the computation. When processing the whole genome with a distance of 6, the public cloud took about 29 hours to complete the job on the 6.8 TB reference hash values. Note that this data preprocessing only incurs a *one-time cost*[4]. Such overheads are completely affordable. For example, Amazon routinely receives terabytes or even a larger amount of data through its Import/Outport service [5].

Seeding performance (every dataset). Preparing the hash values for the 10 million reads turned out to be highly efficient. Even for the edit distance of 6, the keyed hash values for the combined seeds, about 5 GB, were computed from those reads within 7 minutes using a single core. Seeding on such data was also fast. When the distance was 3, the time our prototype took to process all the seeds over the whole genome was merely a little more than 6 minutes, using 20 nodes. When the distance became 6, our approach

[3] CloudBurst extends a seed at every genetic location it matches, rather than drops its read as soon as the read is successfully aligned to a reference substring (within the distance threshold). For the fairness of comparison, our prototype also did all extensions. Note that this is not necessary for the filtering and our approach can achieve a much faster speed when doing the filtering alone.

[4] The reference hashes can be replaced after they are used to process a large amount of data, e.g., 10,000 read datasets (Appendix 7.3).

Table 3. Performance of Preprocessing and Seeding

Reference (# of errors)	Reference data perparation (One-time cost)			Seeding (Every dataset)
	Hash Reference (h:min:sec)	Sort hashed l-mers (h:min:sec) (8 cores/node)	Reference Generated (GB)	(h:min:sec) (8 cores/node)
Chr1 (3)	0:3:33 (1 core)	0:6:54 (5 nodes)	5.7	0:5:29 (5 nodes)
Chr22 (3)	0:0:29 (1 core)	0:2:12 (5 nodes)	0.9	0:5:22 (5 nodes)
Whole Genome (3)	1:1:13 (1 core)	0:23:53 (20 nodes)	79.6	0:6:12 (20 nodes)
Chr1 (6)	0:51:48 (8 cores)	2:39:2 (20 nodes)	558.9	0:13:1 (20 nodes)
Chr22 (6)	0:10:11 (8 cores)	0:17:31 (20 nodes)	88.11	0:5:39 (20 nodes)
Whole Genome (6)	11:20:7 (8 cores)	29:4:43 (30 nodes)	6997.8	1:32:53 (30 nodes)

Table 4. Outsourced Computation

Reference (# of errors)	Workload of Our Private Cloud					Full Workload of CloudBurst			Outsource ratio (%) (*Estimate)
	Hash seeds (h:m:s) (**1 core**)	Extension (**1 core**)		Bandwidth		Workload (h:m:s) (**8 cores/node**)	Bandwidth		
		Time (h:m:s)	Mem (GB)	Up (GB)	Down (MB)		Up (GB)	Down (MB)	
Chr1 (3)	0:1:3	0:1:16	1.85	0.72	11.18	0:31:10 (1node)	0.81	11.32	99.1
Chr22 (3)	0:1:4	0:0:38	1.47	0.72	11.11	0:3:52 (1node)	0.81	11.52	94.5
Whole Genome (3)	0:1:8	0:10:41	6.92	0.72	17.83	0:14:1 (20nodes)	0.81	15.16	99.5 *
Chr1 (6)	0:6:33	0:12:22	3.71	5.07	74.93	0:59:23 (1node)	0.81	998.06	96.1
Chr22 (6)	0:5:26	0:4:9	3.13	5.07	74.96	0:13:29 (1node)	0.81	402.46	91.1
Whole Genome (6)	0:6:1	2:37:27	8.97	5.07	120.71	0:26:43 (30nodes)	0.81	1456.14	97.5 *

understandably ran slower (one hour and a half), given the protection it offers and the seed-combination strategy that moves the cost from the extension to the seeding. In the experiment, we set Java virtual machine's maximum memory heap size to 1.6GB both for the cloud instances running CloudBurst and those working on the seeding tasks in our approach. We found that our approach consumed much less memory than CloudBurst, which kept additional information for extensions.

Extension performance (every dataset). The extension tasks, based upon matched seeds or combinations, were so small that they were all comfortably handled by the single core: even in the case of the whole genome and the distance of 6, our desktop used about two hours and a half to complete the extension. To understand the amount of the workload our approach outsourced to the public cloud, we tried to run CloudBurst on the desktop (using 8 cores). This attempt succeeded when the references were Chr1 and Chr22: in all these cases, at least 91% of the computation were outsourced, even considering the time for preparing the hash values of the seeds. However, running CloudBurst on the whole genome within a single machine turned out to be impossible, which forced us to look at its performance on the public cloud. Specifically, for the distance of 6, CloudBurst spent about 26 minutes on 30 8-core nodes to map all the 10 million reads, whereas our cost on the private cloud were merely 163 minutes using 1 node 1 core, including the time for hashing the seeds. Therefore, we estimated that over 97% workload was offloaded to the public cloud.

Communication overheads (every dataset). We found that the communication overheads of our approach was rather small. The maximum amount of data needed to be uploaded to the public cloud was 5.07 GB (including the hashes of the combined seeds for the 10 million reads), which took only a couple of minutes to transfer on our 40 MBps link. The data downloaded from the public cloud was much smaller, merely 120.71 MB. We can compare this level of overheads with those incurred by having the whole map job done by CloudBurst on the public cloud without any privacy protection: about 0.81 GB data was uploaded in this case and 1.42 GB data needed to be downloaded. Note that our approach only needs to download a relatively small amount of data for matched seed combinations[5] rather than the outcome of the whole computation.

Discussion. The overall computation time (the time spent on the public cloud and the private cloud combined) of our prototype was about 372 CPU hours for the mapping task on the whole genome with an edit distance of 6, which is about 3.5 times as much as that for performing the

[5]Only 5% of the reads belong to human hosts.

whole computation by CloudBurst, without privacy protection at all. Note that all existing privacy-preserving techniques [17, 18, 20, 33, 35] are not even remotely close to achieving this level of performance on the read-mapping problem. On the EC2, this computation only costs $26 (estimated based on the price for reserved Cluster Compute EC2 instance [1], whose computing power is comparable to the cloud nodes we used) . At this expense, one does not need to purchase and maintain a 240-core cluster: a single desktop is sufficient to do the whole work.

5 Related Work

Secure outsourcing of genomic computations. Most of the proposed techniques for secure outsourcing of genomic computations focus on new cryptographic primitives [17,21,35]. For example, a protocol for computing edit distances [17] shares a matrix for dynamic programming to multiple servers, which need to collaborate with each other through homomorphic encryptions and oblivious transfers to calculate each matrix element. This approach was found to need 5 and a half minutes to compute an instance of the size (25, 25) [35]. Another example is the work on optimized SMC for DNA sequence alignment [35], which is designed to leverage the special features of dynamic programming to improve the performance of SMC. Compared with [17], the approach is much more efficient, taking about 14 seconds to complete the above task. Further improvement on SMC [33] can align 2 100-element sequences in 4 seconds. Still, this overhead makes it hard to scale up to the bar of comparing millions of reads with millions to billions of l-mers. Recent developments on this line of research include oblivious automata evaluation [20], which only needs $O(n)$ modular exponentiations to work on the sequences with n elements. This performance, however, still cannot sustain the scale of read mapping. Another recent proposal [14] attempts to "disguise" DNA sequences by scrambling some of its nucleotides to produce multiple versions of the sequence and let multiple servers compute on them. The outcomes of these computations are then analyzed by the client to restore the edit distance between the sequence and an l-mer. The problem with this approach is that the server needs to communicate with the client for every alignment attempt, making its scalability questionable. Fundamentally, those approaches all fail to take advantage of the special features of human genomes, which were utilized in our research to build a simple and practical solution.

Secret-sharing based approaches may bring in new policy challenges: once the data has been shared to multiple parties, the NIH completely loses the control of it, since these parties can work together to restore the data; it is still unclear whether the NIH needs to sign an agreement with each of them, which these parties may want to avoid for liability concerns [3], and if so, what the agreement will look like. This concern is also applied to the approaches such as distributed Smith-Waterman algorithm [53] that decomposes a computation problem into small sub-problems and allocates them to multiple problem solvers, under the assumption that these parties will not collude. Another related approach [57] lets the provider of genomic data replace SNP values with symbols, which allows an untrusted party to perform a program specialization on the sanitized data. The approach assumes that the data provider knows the locations of SNPs in its data, whereas reads do not carry such information before they are mapped onto the reference genome. Also remotely related to our work is the study on the information leaks caused by aligning a query sequence to those in a genomic database [27]. In our research, a similar problem occurs when the public cloud analyzes the frequencies of the hash values of l-mers. This threat, however, was found to be very limited (Section 3.4).

Other secure outsourcing techniques. Early research on secure outsourcing is mainly on delegating cryptographic operations (e.g., modular exponentiations) to a set of untrusted helpers [31, 41]. More recent studies, in addition to secure computing of edit distance, also include the computations such as linear algebra operations [16] and machine-learning tasks [24]. For example, Peer-for-Privacy decomposes a category of data mining algorithms into vector addition steps and distributes them to multiple nodes on a cloud, which can be securely evaluated through a special secret sharing scheme [24]. All these approaches, however, may not be suitable for computing edit distances and also incur a large amount of communication during the computation.

6 Conclusion

In this paper, we propose a suite of new techniques that achieve secure and scalable read mapping on hybrid clouds. Our approach leverages the special features of the read mapping task, which only cares about small edit distances, and of the Cloud, which is good at handling a large amount of simple computation. These features enable us to split the mapping computation according to the seed-and-extend strategy: the seeding stage performs simple computation (exact matching) on a large amount of ciphertext, which is undertaken by the public cloud, and the extension stage involves a very small amount of relatively complicated computation (edit-distance calculation) at the genetic locations found by the matches, which is shouldered by the private cloud. We carefully designed these techniques to move the workload from the private cloud to the public cloud, and thoroughly evaluated their privacy protection and performance. The study shows that the approach, though simple, offers a high privacy assurance and can easily handle the computation of a practical scale.

Acknowledgements

This work is supported in part by the National Science Foundation under Grant CNS-0716292 and CNS-1017782. Also, the FutureGrid testbed used to evaluate our approach is supported in part by the NSF Grant No. 0910812.

References

[1] Amazon EC2 Pricing. http://aws.amazon.com/ec2/pricing/.

[2] Amazon Elastic MapReduce Pricing. http://aws.amazon.com/elasticmapreduce/#pricing.

[3] Amazon Web Services Customer Agreement. http://aws.amazon.com/agreement/.

[4] Amazon Web Services Simple Monthly Calculator. http://calculator.s3.amazonaws.com/calc5.html.

[5] AWS Import/Export. http://aws.amazon.com/importexport/.

[6] CloudBurst on Amazon Web Services. http://aws.amazon.com/articles/2272?_encoding=UTF8&jiveRedirect=1.

[7] Crypto++ 5.6.0 Benchmarks. http://www.cryptopp.com/benchmarks.html.

[8] Human Microbiome Project. http://nihroadmap.nih.gov/hmp/.

[9] International HapMap Project. http://www.hapmap.org/.

[10] SecureGenome. http://securegenome.icsi.berkeley.edu/securegenome/.

[11] The Health Insurance Portability and Accountability Act. http://www.hhs.gov/ocr/privacy/, 1996.

[12] Policy for sharing of data obtained in nih supported or conducted genome-wide association studies (gwas). http://grants.nih.gov/grants/guide/notice-files/NOT-OD-07-088.html#principles, 2007.

[13] Copy Number Variation. http://www.nature.com/scitable/topicpage/copy-number-variation-445, 2008.

[14] R. Akimana, O. Markowitch, Y. Roggeman, and U. L. De. Secure outsourcing of dna sequences comparisons in a grid environment. 2008.

[15] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, October 1990.

[16] M. J. Atallah and K. B. Frikken. Securely outsourcing linear algebra computations. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 48–59, New York, NY, USA, 2010. ACM.

[17] M. J. Atallah, F. Kerschbaum, and W. Du. Secure and private sequence comparisons. In *Proceedings of the 2003 ACM workshop on Privacy in the electronic society*, WPES '03, pages 39–44, New York, NY, USA, 2003. ACM.

[18] M. J. Atallah and J. Li. Secure outsourcing of sequence comparisons. *Int. J. Inf. Secur.*, 4:277–287, October 2005.

[19] R. A. Baeza-Yates and C. H. Perleberg. Fast and practical approximate string matching. In *Proceedings of the Third Annual Symposium on Combinatorial Pattern Matching*, CPM '92, pages 185–192, London, UK, 1992. Springer-Verlag.

[20] M. Blanton and M. Aliasgari. Secure outsourcing of dna searching via finite automata. In S. Foresti and S. Jajodia, editors, *DBSec*, volume 6166 of *Lecture Notes in Computer Science*, pages 49–64. Springer, 2010.

[21] F. Bruekers, S. Katzenbeisser, K. Kursawe, and P. Tuyls. Privacy-preserving matching of dna profiles. Technical Report Report 2008/203, ACR Cryptology ePrint Archive, 2008.

[22] D. F. Conrad, D. Pinto, R. Redon, L. Feuk, O. Gokcumen, Y. Zhang, J. Aerts, T. D. Andrews, C. Barnes, P. Campbell, T. Fitzgerald, M. Hu, C. H. Ihm, K. Kristiansson, D. G. MacArthur, J. R. MacDonald, I. Onyiah, A. W. Pang, S. Robson, K. Stirrups, A. Valsesia, K. Walter, J. Wei, C. Tyler-Smith, N. P. Carter, C. Lee, S. W. Scherer, and M. E. Hurles. Origins and functional impact of copy number variation in the human genome. *Nature*, 464(7289):704–712, Oct. 2009.

[23] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51:107–113, January 2008.

[24] Y. Duan, N. Youdao, J. Canny, and J. Zhan. P4p: Practical large-scale privacy-preserving distributed computation robust against malicious users abstract. In *Proceedings of the 19th USENIX Security Symposium*, Washington, DC, August 2010.

[25] C. Dwork. Differential privacy. In *in ICALP*, pages 1–12. Springer, 2006.

[26] B. Furht. Cloud computing fundamentals. In B. Furht and A. Escalante, editors, *Handbook of Cloud Computing*, pages 3–19. Springer US, 2010.

[27] M. T. Goodrich. The mastermind attack on genomic data. *Security and Privacy, IEEE Symposium on*, 0:204–218, 2009.

[28] D. Gusfield. *Algorithms on strings, trees, and sequences : computer science and computational biology*. Cambridge Univ. Press, January 2007.

[29] F. Hach, F. Hormozdiari, C. Alkan, F. Hormozdiari, I. Birol, E. E. Eichler, and S. C. Sahinalp. mrsFAST: a cache-oblivious algorithm for short-read mapping. *Nature Methods*, 7(8):576–577, Aug. 2010.

[30] Heeney C, Hawkins N, de Vries J, Boddington, and P, Kaye J. Assessing the Privacy Risks of Data Sharing in Genomics. . *Public Health Genomics*, 2010.

[31] S. Hohenberger and A. Lysyanskaya. How to securely outsource cryptographic computations. In J. Kilian, editor, *Theory of Cryptography*, volume 3378 of *Lecture Notes in Computer Science*, pages 264–282. Springer Berlin / Heidelberg, 2005.

[32] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V. Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS Genet*, 4(8):e1000167+, 2008.

[33] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *Proceedings of the 20th USENIX Security Symposium*, San Francisco, CA, August 2011.

[34] K. B. Jacobs, M. Yeager, S. Wacholder, D. Craig, P. Kraft, D. J. Hunter, J. Paschal, T. A. Manolio, M. Tucker, R. N. Hoover, G. D. Thomas, S. J. Chanock, and N. Chatterjee. A new statistic and its power to infer membership in a genome-wide association study using genotype frequencies. *Nature Genetics*, 41(11):1253 – 1257, 2009.

[35] S. Jha, L. Kruger, and V. Shmatikov. Towards practical privacy for genomic computation. *Security and Privacy, IEEE Symposium on*, 0:216–230, 2008.

[36] W. J. Kent. BLAT: The BLAST-Like Alignment Tool. *Genome Research*, 12(4):656–664, April 2002.

[37] B. Langmead, C. Trapnell, M. Pop, and S. Salzberg. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biology*, 10(3):R25, 2009.

[38] H. Li and N. Homer. A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics*, 11(5):473–483, September 2010.

[39] J. E. Lunshof, R. Chadwick, D. B. Vorhaus, and G. M. Church. From genetic privacy to open consent. *Nature reviews. Genetics*, 9(5):406–411, May 2008.

[40] E. R. Mardis. The impact of next-generation sequencing technology on genetics. *Trends in Genetics*, 24(3):133 – 141, 2008.

[41] T. Matsumoto, K. Kato, and H. Imai. Speeding up secret computations with insecure auxiliary devices. In *Proceedings of the 8th Annual International Cryptology Conference on Advances in Cryptology*, pages 497–506, London, UK, 1990. Springer-Verlag.

[42] S. A. McCarroll and D. M. Altshuler. Copy-number variation and association studies of human disease. *Nature genetics*, 39(7 Suppl), July 2007.

[43] NSF. Award abstract #091081 - futuregrid: An experimental, high-performance grid test-bed, 2009.

[44] O. OMalley and A. C. Murthy. Winning a 60 second dash with a yellow elephant. Technical report, Yahoo! Inc., 2009.

[45] J. e. a. Qin. A human gut microbial gene catalogue established by metagenomic sequencing. *Nature*, 464(7285):59–65, March 2010.

[46] S. Sankararaman, G. Obozinski, M. I. Jordan, and E. Halperin. Genomic privacy and limits of individual detection in a pool. *Nat Genet*, 41(9):965–7, 2009.

[47] M. C. Schatz. CloudBurst: highly sensitive read mapping with MapReduce. *Bioinformatics*, 25(11):1363–1369, 2009.

[48] N. J. Schork, S. S. Murray, K. A. Frazer, and E. J. Topol. Common vs. rare allele hypotheses for complex diseases. *Current opinion in genetics & development*, 19(3):212–219, June 2009.

[49] S. C. Schuster. Next-generation sequencing transforms today's biology. *Nature Methods*, 5(1):16–18, December 2007.

[50] A. Smith, Z. Xuan, and M. Zhang. Using quality scores and longer reads improves accuracy of solexa read mapping. *BMC Bioinformatics*, 9(1):128, 2008.

[51] A. D. Smith, W.-Y. Chung, E. Hodges, J. Kendall, G. Hannon, J. Hicks, Z. Xuan, and M. Q. Zhang. Updates to the RMAP short-read mapping software. *Bioinformatics*, 25(21):2841–2842, 2009.

[52] L. Stein. The case for cloud computing in genome informatics. *Genome Biology*, 11(5):207+, May 2010.

[53] D. Szajda, M. Pohl, J. Owen, and B. G. Lawson. Toward a practical data privacy scheme for a distributed implementation of the smith-waterman genome sequence comparison algorithm. In *NDSS*. The Internet Society, 2006.

[54] C. Trapnell and S. L. Salzberg. How to map billions of short reads onto genomes. *Nature biotechnology*, 27(5):455–457, May 2009.

[55] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik. Privacy preserving error resilient dna searching through oblivious automata. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, pages 519–528, New York, NY, USA, 2007. ACM.

[56] R. Wang, Y. F. Li, X. Wang, H. Tang, and X. Zhou. Learning your identity and disease from research papers: Information leaks in genome wide association study. In *CCS '09: Proceedings of the 15th ACM conference on Computer and communications security*, pages 534–544, New York, NY, USA, 2009. ACM.

[57] R. Wang, X. Wang, Z. Li, H. Tang, M. K. Reiter, and Z. Dong. Privacy-preserving genomic computation through program specialization. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*, pages 338–347, New York, NY, USA, 2009. ACM.

7 Appendix

7.1 Read Anonymization and Re-Identification

Anonymizing read data. Different from the microarray data that targets a specific set of SNPs, sequencing reads are randomly drawn from a human genome, which consists of 6 billion nucleotides. In a dataset with 1 million sequences of 100-bp long, about 40% of these reads carry SNPs, typically, one on each of them. These SNPs, roughly 0.03 of the total 14 million SNPs, can be viewed as randomly picked out from the whole SNP set. This ratio can become even lower when it comes to human microbiome sequencing data [8], which has about 1-8% of the reads contaminated from the respective human host. Given that each random sample (all SNPs on the reads from one individual) is small relative to the total number of SNPs, the overlapping between two different persons' sequence datasets, in terms of the SNPs they share, is often not significant. Therefore, the question becomes that once we aggregate multiple persons' read datasets, each carrying a different set of SNPs, whether the resulting mixture can still be used to re-identify these individuals. We can make this identification attempt even more difficult to succeed by randomly adding to the mixture *noise* reads, which are randomly sampled from the reference genome with its SNP sites randomly

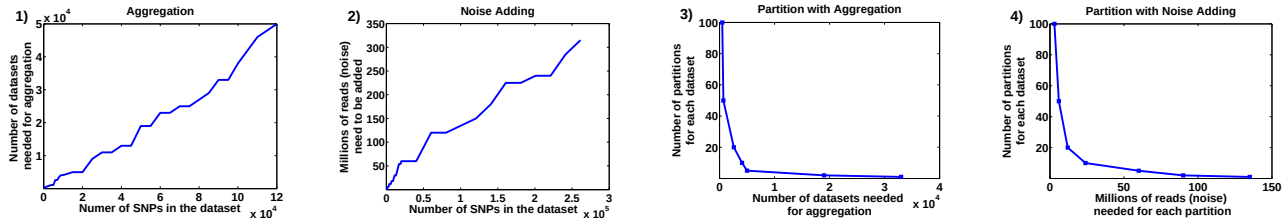

Figure 5. Effectiveness of simple mitigation techniques: 1)aggregation, 2)noise adding, 3)data partition after aggregation, and 4)data partition after noise adding.

set to major/minor alleles according to the known allele frequencies in human population (e.g. taken from the HapMap project [9]), and/or by partitioning an individual's dataset into multiple subsets to let the public cloud process them separately.

The re-identification threat. Our research shows that these techniques are still insufficient for protecting the read data from the adversary with access to a reference population and a DNA sample from the victim. Specifically, consider a dataset whose reads are sampled from a population (referred to as the case group). We first estimate the allele frequencies in the case group by aligning these reads to the reference genome: if a SNP site k has been covered by m reads, and i of them have the major allele and the rest carry the minor allele, the major allele frequency is calculated as $f_k = \frac{i}{m}$. Note that this frequency often deviates from the SNP's real major-allele frequency in the population, simply because not everyone's SNP k has been sampled. Actually, it might well be that many SNP sites are not covered by any read and therefore have frequencies of zero. The case group here describes a group of human subjects whose reads are aggregated into the sequence dataset, as well as a mixture of real and fake humans, when artificial reads are added to the dataset as noise. In either case, our objective is to determine whether or not an individual (the testee) is present in the case group from the allele frequencies we observe. To this end, we also need a *reference group*, for example, the HapMap population [9] whose allele frequency for each SNP k, F_k, is public knowledge, and a sequence of allele pairs[6] from the testee, one pair for each of the SNP site k whose major-allele frequency Y_k can be 0 (two minor alleles), 0.5 (one major and one minor) or 1 (two major). Based upon such information, we analyzed anonymized read data using a statistic proposed by Homer, et al. [32]:

$$D_k = |Y_k - F_k| - |Y_k - f_k| \tag{3}$$

Assuming that the distributions of SNPs' allele frequencies in the case and reference populations are identical, the sum of D_k across all independent SNP k will have a normal distribution, whose mean becomes zero when the testee is not in the case/reference groups, and significantly larger than zero when she is a case. By testing this statistic on the null hypothesis: "the testee is not within the case group", we assessed the level of privacy protection that different anonymization techniques are able to offer. Note that although this statistic is well-known to be effective on the aggregated microarray data [12, 32], the vulnerability of the anonymized sequence data to such a re-identification attack has not been investigated before.

The setting of our study. To evaluate the re-identification risk in outsourcing anonymized read data, we performed a series of hypothesis tests on such data under four scenarios: 1) aggregation only, 2) noise-adding only, 3) aggregation and then data partition and 4) noise-adding and then data partition. All the genomic sequences used in our study were randomly sampled from the reference genome: when a sampled read covered a SNP site, its allele was randomly picked according to the major allele frequency of the site in the YRI population, as provided by the HapMap [9]. In this way, we acquired the realistic sequencing reads from a large group of simulated people. Our research utilized published 3,138,397 SNP sites of the YRI population in the HapMap dataset. We consider an anonymized dataset with 100-bp reads to be not secure if a sequence donor for the dataset has a probability of at least 0.1 to be identified with a confidence level no less than 0.99 (i.e. a false positive rate no more than 0.01). Our analysis aims at determining the necessary condition, e.g, the minimum number of the *personal datasets* (the dataset of an individual's reads) needed to be aggregated or the noise reads needed to be added, to protect the person from being re-identified through her N SNPs in the dataset.

Our findings. The outcomes of our evaluation study are presented in Figure 5. In the leftmost panel of the figure, we show the cost for the aggregation strategy. The simple test statistic in Equation 3 was found to be able to pose an alarming level of threat to the DNA donors, exactly as it does to the GWAS participants through their aggregated allele frequencies derived from microarray data: as illus-

[6] A human inherits genes from both of his/her parents and therefore has two alleles at each SNP site. For a case individual, his/her two alleles appear on two different reads.

trated in the panel, to cover the identity information disclosed by the N SNPs from each donor (x-axis), a large number of personal datasets (each with N SNPs) have to be aggregated (y-axis). As an example, consider a human microbiome sequencing dataset that contains 10 million reads with 3% of human contamination. These human reads cover about 100,000 SNPs, and therefore, according to the figure, need an aggregation of at least 38000 personal datasets of the same size to secure, which. This amount of data cannot be afforded by even the largest microbiome project.

Noise adding is another way to reduce the privacy risks in outsourcing read data. In our study, we generated noise reads covering major/minor alleles at randomly-chosen SNP sites and evaluated the re-identification power achievable over the personal dataset including these reads. The middle-left panel in Figure 5 shows the minimum number of noise reads (y-axis) that are required to secure the dataset with N SNPs from a donor (x-axis). Our study shows that the number of the required noise reads grows linearly with regards to N. For example, at least 140 million noise reads need to be added in order to secure a human microbiome sequencing dataset with 10 million reads covering about 100,000 SNPs.

We further studied the strategies that partition the datasets after they were anonymized through aggregation or noise adding. All our analyses were performed on the personal dataset that contained 10 million reads and covered about 100,000 human SNP sites, a large case in human microbiome sequencing. The middle-right panel in Figure 5 shows the number of partitions needed (y-axis) to secure a dataset aggregated over different numbers of personal datasets (x-axis), and the rightmost panel demonstrates the number of required partitions (y-axis) vs. the number of noise reads being added (x-axis). As illustrated by the figure, when it is possible to partition an aggregated dataset into 100 subsets for the public cloud to process independently, the dataset should be built from at least 500 personal datasets, or carry at least 100 million noise reads (1 million noise reads per subset) to stay safe, which are better than aggregation or noise adding alone, though the overheads are still significant (particularly when it comes to other read datasets with higher levels of SNPs). Moreover, data partition could bring in large communication overheads, because each subset needs to be transferred to the public cloud separately. It is also less clear how to prevent the public cloud from linking different subsets together (e.g. based on the those who submit the jobs): when this happens, the cloud can aggregate the subsets for the re-identification.

7.2 Identification Attacks on Combined Seeds

We demonstrate that the l-mer based near-optimal statistics have no power at all, which indicates that the frequency analysis on the keyed-hash values of l-mers does not offer sufficient information for a re-identification attack. Below, we show that the Homer-like statistic cannot achieve a higher power on the combined seeds than on the continuous seeds. The similar analysis can be applied to the log likelihood ratio test. Consider a continuous seed (24 bps) consisting of two consecutive 12-mers, one of which contains a SNP site (denoted by α_1). Because most of the 100-bp windows in the human genome contain at most one SNP site, totally there are $100 - 12 = 88$ combined seeds within the same 100-bp window that contain the SNP site. All of them include α_1, thus can be denoted as $\alpha_1||\alpha_i$ ($i = 2, ..., 89$), and one of them is the continuous seed (denoted as $\alpha_1||\alpha_2$). Let $B_{k(i)}$ ($i = 2, ..., 89$) be the bins for combined seeds $\alpha_1||\alpha_i$ when α_1 carries a major allele, and $B_{k'(i)}$ be the bins for $\alpha_1||\alpha_i$ when α_1 carries a minor allele. Because all these seeds involve the same SNP site on the 12-mer α_1, all the related seeds from the testee's genome must all carry the same allele, major or minor. Then, the numbers of l-mers in the bins $B_{k(i)}$ and $B_{k'(i)}$ ($\bar{f}_{k(i)}$ and $\bar{f}_{k'(i)}$, respectively) are equally deviated from their expected counts ($\bar{F}_{k(i)}$ and $\bar{F}_{k'(i)}$, respectively) as compared to the numbers of l-mers in the bins of the continuous seeds ($\bar{f}_{k(2)}$ and $\bar{f}_{k'(2)}$) deviated from their expected counts ($\bar{F}_{k(2)}$ and $\bar{F}_{k'(2)}$). Hence, we have $\sum_i(|\rho_{k(i)} - \bar{F}_{k(i)}| + |\rho_{k'(i)} - \bar{F}_{k'(i)}|) \approx 88 \times (|\rho_{k(2)} - \bar{F}_{k(2)}| + |\rho_{k'(2)} - \bar{F}_{k'(2)}|)$, and $\sum_i(|\rho_{k(i)} - \bar{f}_{k(i)}| + |\rho_{k'(i)} - \bar{f}_{k'(i)}|) \approx 88 \times (|\rho_{k(2)} - \bar{f}_{k(2)}| + |\rho_{k'(2)} - \bar{f}_{k'(2)}|)$. Therefore, the Homer-like statistic on the bins of all combined seeds will become: $\bar{T}_{com} \approx 88 \times \bar{T}$ where $\bar{T}$ is the Homer-like statistic on the bins of continuous 24-bp seeds (as defined in Equation 1). This implies that, no matter the testee is a case individual or not, the test statistic on all combined 12-bp seeds ($\bar{T}_{com}$) is a constant (88) times larger than the test statistic on the continuous 24-bp seeds $\bar{T}$. As a result, at the same confidence level, $\bar{T}_{com}$ cannot achieve higher power than $\bar{T}$. Since we have shown $\bar{T}$ has little re-identification power, $\bar{T}_{com}$ should not either.

7.3 Correlation Analysis

Here we show that identification of correlated hash values is very hard in practice[7] and attempts to do so can be easily defeated. This challenge (to the adversary) comes from the fact that she only observes the hash values for a small portion of a donor's genome: by far the largest dataset from an individual contains no more than 10 million reads (of 100 bps long each), only $\gamma = 1/150$ of all the 24-mers on her genome. To understand what the adversary can do, let us first consider the simplest case, when two 24-mers are completely correlated. This happens when they contain al-

[7]Remember that each hash value in a bin cannot be uniquely linked to an l-mer and therefore, the correlations among hashes cannot be identified through l-mers.

leles of the same SNP: if both share the same allele, they are positively correlated; otherwise, if one contains the major allele and the other contains the minor allele, they are negatively correlated. To detect such correlations, the adversary needs to conduct a co-occurrence statistical test to find out whether the hash values of these 24-mers always appear or disappear together in different donors' seed-hash datasets. To defeat this attack, we can aggregate the reads from 20 individuals (sampled from 40 DNA sequences, 2 from each) in one read-mapping task. After the aggregation, in a single aggregated dataset, no SNP site likely contains only minor alleles: the probability for this to happen is below $2^{-40} \approx 10^{-12}$, since the minor allele frequency is below 0.5. In other words, l-mers with major alleles are always carried by some case individuals, regardless of the presence or absence of related l-mers in the seed-hash dataset the adversary sees. As a result, the test will fail on the pairs of 24-mers that both contain major alleles or contain one major and one minor allele. For the pair of 24-mers both containing a minor allele, the adversary can confidently conjecture they are correlated only if she observes both 24-mers in multiple aggregated datasets, because the probability of observing a pair of minor-allele-containing 24-mers in a specific dataset by chance is not small. Assuming there are N SNP sites in the human genome ($N > 10^6$), the random probability of observing a pair of minor-allele-containing 24-mers in n aggregated datasets is approximately $P \sim N^2 \times l^2 \times (\gamma^2 \times t)^n$ (where t is the minor allele frequency and $t < 0.5$), indicating it requires $n \geq 4$ for $P \ll 1$. On the other hand, the probability of observing a highly correlated pair of minor-containing 24-mers in 4 aggregated datasets is very small, unless there are many aggregated datasets to be analyzed. For example, if $M = 500$ aggregated datasets (i.e., $20 \times 500 = 10,000$ human genome reads datasets) are analyzed, the probability of observing 4 aggregated datasets that contain both minor-allele-containing 24-mers can be estimated by a binomial distribution: $P < 500^4 \times (\gamma^2/2)^4 \approx 10^{-8} \ll 1$. Even considering there are many potentially correlated SNP pairs (e.g., close to the total number of SNPs, $N \approx 10^7$), the expected number of confidently assigned l-mer pairs is very small. Therefore, the test will also fail on the pairs of 24-mers containing the same minor allele.

Alternatively, the adversary may attempt to correlate two hashed 24-mers (which can be only partially correlated) at two specific genetic locations through their *relative frequencies* across multiple samples. In other words, she wants to know whether the frequency of one of these two hashes changes with that of the other over different *samples*. Here a sample we refer to includes one individual's read data that indeed contains the substrings at the locations of these two 24-mers, and the frequency of the 24-mer is calculated over multiple such samples. What we want to understand here is whether the adversary can acquire enough samples to establish a reliable correlation between the 24-mers. Remember that the probability a read dataset includes the substring at a specific location is only $\gamma = 1/150$. The adversary needs a set of 10 such samples to calculate the frequency of the 24-mer (at the location) at the precision of one decimal digit and multiple such sets to correlate two 24-mers. Due to the presence of 2% sequencing errors, the best correlation coefficient the adversary can get between a pair of completely correlated 24-mers (at two specific locations) is within ± 0.5 because 40% of the instances of the 24-mers contain an error at 2% error rate per nucleotide, and therefore their hash values will become incorrect and their presences will not be observed in the seed-hash datasets. As a result, the adversary has to carry out a correlation test on at least 15 sets (i.e. 150 samples) to obtain a confidence level (P-value) of 0.05 based on the table of critical values for Pearson's correlation coefficient (other statistics yield similar results). Note that the confidence level becomes even lower when those 24-mers are *not* completely correlated. Therefore, assume the adversary has collected M samples, the probability for a pair of completely correlated 24-mers to be both observed in at least 150 (out of M) samples can be estimated by a normal approximation of the binomial distribution with $\mu = \sigma^2 = \gamma \times M$. When $M = 10,000$ (that is, $10,000$ read datasets), the probability of getting those 150 samples from those datasets is negligible ($\approx 10^{-28}$) even when we consider that a total of $24 \times 14 \times 10^6 = 3.4 \times 10^8$ 24-mers are subject to this correlation analysis (there are 14 million SNP sites in the human genome, each associated with 24 24-mers). For the 24-mers not completely correlated, for example, those containing different SNPs, the probability to get 150 samples in 10,000 datasets is even lower, because the chance to have both 24-mers in one sample becomes $(0.6\gamma)^2$. Therefore, even when we consider the 2-combinations of all 3.4×10^8 24-mers, the probability to correlate any two of them using 10,000 read datasets is well below 10^{-12}. This is the best chance that the adversary can correlate a single pair of 24-mers (which is not enough for a re-identification). If such a risk is acceptable to the data owner, what she can do is re-hashing the reference genome with a new secret key every 10,000 datasets. The cost for this update is small: SHA-1 took about 5,440 minutes of CPU time to hash the whole genome on a 8-core desktop (2.93 GHz Interl Xeon) used in our study and produced about 6.8 TB data; the average overheads are merely 40 seconds CPU time and 700 MB data transfer for each of the 10,000 datasets.