Efficient Genome-Wide, Privacy-Preserving Similar Patient Query based on Private Edit Distance

Xiao Shaun Wang[1], Yan Huang[2], Yongan Zhao[2], Haixu Tang[2], XiaoFeng Wang[2], and Diyue Bu[2]

[1]University of Maryland wangxiao@cs.umd.edu

[2]Indiana University, Bloomington {yh33,yongzhao,hatang,xw7}@indiana.edu

ABSTRACT

Edit distance has been proven to be an important and frequently-used metric in many human genomic research, with Similar Patient Query (SPQ) being a particularly promising and attractive example. However, due to the widespread privacy concerns on revealing personal genomic data, the scope and scale of many novel use of genome edit distance are substantially limited. While the problem of private genomic edit distance has been studied by the research community for over a decade [5], the state-of-the-art solution [30] is far from even close to be applicable to real genome sequences.

In this paper, we propose several private edit distance protocols that feature unprecedentedly high efficiency and precision. Our construction is a combination of a novel genomic edit distance approximation algorithm and new construction of private set difference size protocols. With the private edit distance based secure SPQ primitive, we propose GENSETS, a genome-wide, privacy-preserving similar patient query system. It is able to support searching large-scale, distributed genome databases across the nation. We have implemented a prototype of GENSETS. The experimental results show that, with 100 Mbps network connection, it would take GENSETS less than 200 minutes to search through 1 million breast cancer patients (distributed nation-wide in 250 hospitals, each having 4000 patients), based on edit distances between their genomes of lengths about 75 million nucleotides each.

Categories and Subject Descriptors

K.6.0 [**Management of Computing and Information Systems**]: Security and Protection

Keywords

Secure Computation; Genomic Computation; Edit Distance

1. INTRODUCTION

Consider a physician seeking the best clinic decision for her patients. Invaluable to the effort is the information how other similar patients respond to different therapies. As today's sequencing technologies have cut the cost of whole genome sequencing down to roughly $1000 per person [52], it is highly anticipated that genome-based *Similar Patient Queries* (SPQ) will be used to identify similar patients from a large number of Electronic Medical Records, through a health information exchange (HIE) system such as *PatientsLikeMe* (a patient powered resesearch network [1]), or other emerging systems like the Memphis HIE, Indiana HIE and Illinois HIE. Among the indicators of genetic similarity, *edit-distance* is one of the most important metrics, which is very useful in the biomedical research for the diagnosis and treatment of cancer, Alzheimer's disease, Schizophrenia, etc [44, 49, 22, 51].

Genome-wide Secure SPQ. Standing in the way of deploying a national-scale, genome-wide SPQ system, however, is the privacy and liability concerns in the dissemination of such data. While unauthorized disclosure of personal genome data could cause serious harm to patients, such as denial of insurance, employment and education opportunities or blackmail [29], getting proper authorizations from millions of patients to share their data is not easy, due to its complicated procedure. Further, searching disease data solely relying on signed agreements can be less realistic in the near future, particularly when it comes to the secondary use (e.g., biomedical research). As a result, in the absence of scalable techniques that enable data use without exposing its content to unauthorized parties, the chance for any SPQ system to be deployed in practice is remote, at least in the near future.

Addressing such privacy challenges in supporting SPQ over distributed genomic datasets seems right up the alley of *Secure Multi-Party Computation* (SMC). Despite continuous performance improvements of secure computation in recent years, the scalability and performance of the state-of-the-art edit-distance based SPQ is still far from usable in supporting SPQ queries: the most efficient SMC implementation can only compute the edit distance between two sequences of a few thousands of base pairs, at a cost of hours of computing time and tens of gigabytes of bandwidth consumption [30]. This is completely off the scale expected for a nation-wide SPQ system.

Secure query at the national scale. To enable patients to benefit from the soon-to-be-available, enormous amount of clinic genomic data, we propose a suite of novel techniques to offer secure SPQ based on the edit distance metric. Our approach, called GENSETS (Genome-wide, Secure Patient Search), is capable of searching 250 hospitals each containing 4000 patients (totally 1 million patients) across the nation within 200 minutes, by securely thresholding the edit distances over real genome data from breast cancer patients (Section 4.1).

Underpinning GENSETS are a few key insights that enables a simple and effective edit-distance based SPQ design. First, we observe a unique feature in human genome sequences and exploit it in developing a highly accurate approximation of edit distance be-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

CCS'15, October 12-16, 2015, Denver, CO, USA
© 2015 ACM. ISBN 978-1-4503-3832-5/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2810103.2813725.

tween genomes. More specifically, variations across the genome sequences of two average human individuals (even those associated with different genetic diseases) are dominated by nuleotide substitutions, with sporadic insertions and deletions scattered across the genome. Second, leveraging a public *reference genome* (so the variations of one's private genome from the reference genome can be locally computed) and pre-computed private variations, the edit-distance between two private genomes can be efficiently approximated using *set difference size* from the (much shorter) private variations. Third, the size of private set difference can be securely approximated (without ever computing the private set difference) using probabilistic algorithms; furthermore, secure thresholding on the size of private set difference can be computed even more efficiently without ever computing even the size of set difference.

Combining those key observations above, we convert the problem of private edit distance into a much simpler problem of approximating the *size of set difference* (Section 3.2). We showed, through running our prototype in realistic continental network environment, that the edit distance between the whole-genomes of two persons can be securely calculated in less than 40 seconds, at an error rate of 1.5%; while comparing the edit distance of two persons' whole-genomes with a threshold value can be done even faster, consuming less than 0.9 seconds to achieve 0.01% false positive/negative rate.

This secure SPQ primitive (based on secure edit-distance) can potentially be deployed to support two-stage queries, in which hospitals will group their patients into clusters so that the first stage query identifies (by computing private edit distance between the query and the cluster center, which is a patient in the cluster) candidate clusters that contain similar patients; while the second stage only searches similar patients in those candidate clusters.

We implemented the secure SPQ primitive and evaluated our approach over a real genome dataset consists of 105 breast cancer patients (data obtained from dbGaP/TCGA with IRB approval). We run SPQ experiments over a cross-country network. With a 100 Mbps network connection, GenSets can accurately execute a SPQ query in less than 200 minutes to search through 1 million patients distributed in 250 hospitals (assuming each hospital has 4000 patients' records and that at most 5 candidate hospitals are selected to continue to the second stage). This result shows that our techniques have moved secure SPQ, one of the most important application of HIE, close to practice use.

Contributions. The contributions of the paper include:

- *New techniques.* We developed a new approach to realize secure SPQ based on whole-genome edit-distance, attaining unprecedented high performance. We achieved this by exploiting intrinsic features of human genome data and efficient probabilistic approximation algorithms. Specifically, we propose an efficient, scalable algorithm to approximate edit-distance for human genomes (Section 3.2), two efficient, probabilistic private set difference size protocols and an efficient, probabilistic private set difference size thresholding protocol (Section 3.4). These, together with a new two-step SPQ search scheme, moves the privacy-preserving SPQ closer to the national scale than it has ever been.

- *Implementation and evaluation.* We implemented our design and evaluated it in a large-scale experiment using realistic genome dataset. The experiments demonstrate a promising prospect of deploying privacy-preserving SPQ systems. Our system will be made open source to the community at `https://github.com/SPQ-EditDistance/code`.

A Note on Security. The proposed approach in this paper faces the security concern raised by Feigenbaum et al. [23], since we rely on the two parties to locally compute their sketches using public hash functions with public randomness. To get around this issue, we adopted a weaker notion of security defined with respect to a modified ideal world execution (see Section 3.1), where, aside from the approximation outcome, the randomness used in constructing the sketches is also revealed. It remains an open question whether our approach is secure (or insecure) with respect to the standard definition of ideal world execution (where the randomness is not revealed).

2. BACKGROUND

Genetic variations and SPQ. The human genome includes two complementary strands, with 3 billion DNA bases each. Each unit on the strand is a nucleotide (A, T, C or G). Between two randomly-selected individuals, over 99% of their nucleotides are identical, with the rest different due to genetic variations. The most common variation involves only a single nucleotide, which can be either a major allele "0", or a minor one "1". Such a variation is called *Single Nucleotide Polymorphism* (SNP). About 50 million nucleotides in human genome are marked as SNPs according to dbSNP [47], while two average individuals' genomes typically differ in 4-5 million variation sites.

The DNA data produced by the sequencer are in the form of a large number of short sequences, which are later *assembled* into a whole sequence by aligning each short sequence with a public *reference genome*. The differences between the sequence and the reference, including the nucleotide(s) changed, inserted or deleted at different genetic positions, are documented in a VCF (Variation Call Format) file. A genome-wide SPQ actually happens on the VCF representations of two genome sequences. Note that restricting the comparison to only a set of genetic markers for certain diseases often leads to inferior medical decisions, because, on the one hand, the state-of-the-art understanding of the association between diseases and genetic variations are dynamically improving, and on the other hand, many other variations, which are not part of the disease's genetic markers, could also affect a treatment decision (e.g., patients' reaction to a therapy, known as the *pharmacogenomic markers* [22]).

Secure Computation. The goal of secure computation is to allow several parties to jointly compute a function over secret input data supplied by each party, without using a trusted third party. The theory of secure computation is able to offer a security guarantee as strong as what can be achieved with a trusted third party, i.e., absolutely no information leak beyond what can be inferred from the desired outcome of the function. Since its inception in early 1980s [53, 28], many constructions have been proposed, reducing the security guarantee either to the certain computational hardness assumptions [54, 39], to the dominance of honest participants [12, 8], or to the availability of a source of correlated randomness [9, 10]. In this paper, our construction is built and tested with the garbled circuit protocol. More recently, many cryptographic [41, 36, 46, 11, 55] and implementational [41, 30, 40] optimizations have been proposed that significantly improve the state-of-the-art of garbled circuit protocols.

Threat model. We focus on the honest-but-curious (a.k.a. semi-honest) model, where the parties are trusted to always follow the protocol specification but would do arbitrary (efficient) side computation in an attempt to violating the security of the system. This model makes sense in many real-world applications as launching an instance of a secure computation protocol alone already requires substantial level of trust among the participants, e.g., through a mutual (but weaker flavor of) agreement. Given a honest-but-

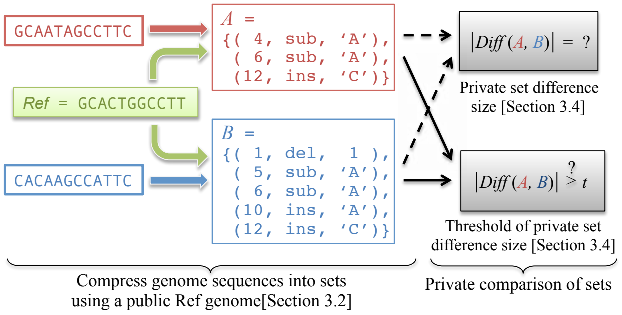

Figure 1: Secure protocols of human-genome edit distances

curious protocol, Huang et al. [31] give a highly efficient dual execution protocol that leaks only one extra bit of information in presence of fully malicious attackers, while many cryptographic techniques [35, 38, 33, 45, 19, 26] could be automated to strengthen semi-honest protocols to work with active adversaries.

Feigenbaum et al. [23] studied the general problem of secure approximation. They pointed out that the standard definition of security might not be always achievable if a sketching algorithm is not fully executed using generic secure computation. The security of our approach relies on an additional assumption (which is also used by Lindell and Pinkas [37]) that learning the randomness used by the sketch algorithm in addition to the approximation results does not provide non-negligible advantage to the adversary in breaking the system.

3. DESIGN AND IMPLEMENTATION

GenSets consists of a highly accurate approximation of edit distance between human genomes (Section 3.2), efficient private set difference size protocols and an efficient private set difference size thresholding protocol (Section 3.3). We will also discuss genome-wide clustering and two-step secure SPQ infrastructure in Section 3.5. We begin our description with an high level overview.

3.1 Overview

The SPQ Primitive. GenSets is built around primitive SPQ protocols based on secure edit distance, which is arguably one of the most important biological similarity indicators [49]. The *edit distance* between sequences A and B is defined as the minimum number of edits (insertion, deletion, or substitution of a single character is counted as *one edit*) to change A into B. Edit distance computation over generic input sequences requires $O(n^2)$ time, which does not scale well on large inputs such as human's whole genome sequences. Computing edit distance is especially challenging in the privacy-preserving setting: the state-of-the-art protocol computing the distance between two sequences of lengths only 2K and 10K in a Giga-bits LAN setting requires more than 3.5 hours and 38 GB of network traffic [30]. This is also the largest problem instance that has ever been attempted in privacy-preserving setting thus far.

We propose several new secure protocols for edit distance. Our protocols are order-of-magnitude more efficient — 20 seconds to securely compute (with an error rate of less than 1%) the edit distance between two whole-genomes (each containing roughly 3 billion bases) and merely 0.1 second to securely threshold (with reasonable false positive/negative rates) the edit distance between two whole-genomes. Our protocol does introduce errors, but only at very limited scale: we have shown through experiments that errors resulted from the secure edit distance protocol applied to real human genomes are within 0.25∼0.5% of the true values; while the false positive and false negative rate of our private edit distance thresholding protocol running on our realistic human genome dataset are within 0.01%.

Our high level idea is illustrated in Figure 1. First, each party will agree on a public reference genome Ref and independently compress local genomes with respect to Ref (by recording the minimum sequence of edits to derive itself from Ref). As a toy example, given the public reference genome Ref to be GCACTGGCCTT, the genome sequence A=GCAATAGCCTTC can be denoted as a set A of operations, $\{(4, sub, \mathtt{A}), (6, sub, \mathtt{A}), (12, ins, \mathtt{C})\}$, i.e., the minimum edits to convert the sequence Ref into A. Due to the information redundancy in human genomes, this step can typically compress a genome string representation of about 3 billion base pairs into roughly 5 million edits (stored in a VCF file). The key insight is that the edit distance between two human genomes A and B can be approximated both efficiently and accurately, through comparing only the VCF file representation of their edits from a single common Ref. Note that in our simplified toy example, the set of edits (desirably) contains only single-character operations, while the VCF file for a real genome will contain multi-character operations. Section 3.2 will elaborate the detailed algorithm to handle these complications and an extensive, empirical study of the accuracy of this approximation applied to human genomes.

Once the sequence of edits in a VCF file is converted into a set of single-character edits, the edit distance of two genomes can be approximated by the size of the symmetric difference between the two sets of single-character edits associated with the two genomes. Note that for the purpose of handling whole-genomes, each single-character-edit set typically contains 8∼10 million edits. This is, unfortunately, still a scale too large to be efficiently handled by existing private set difference protocols. To this end, we propose a highly efficient private set difference size protocol exploiting the idea of probabilistic sketches (Section 3.3). Additionally, observing that the most frequent computation in our SPQ application is actually comparing an edit distance to a threshold value, we introduce a secure set difference size thresholding protocol which runs another order-of-magnitude faster than (our already fast) private set difference size protocol. All these protocols related to private set difference size are generic and readily composable with other secure computation protocols, hence maybe of independent interest.

Infrastructure. Built on top of the primitive is a secure SPQ infrastructure, as illustrated in Figure 2. Consider that a physician makes an SPQ for her breast cancer patients across hundreds of hospitals. Before the query happens, each hospital can pre-processed their data, grouping its patients' DNA sequences into a few clusters. For each cluster, a synthesized sequence that represents the center of the cluster is generated to support the query.

With the pre-computed clusters, the whole SPQ happens in two stages. In the first stage, the querier (the physician's secure SPQ client) runs the secure SPQ primitive with each hospital who supplies just the cluster centers, in order to identify all the hospitals that could have the similar patients (i.e., has at least one cluster center close to the query). Then, the second stage could be launched between the querier and all candidate hospitals identified as a result of the first stage, to securely scan through all patients in these hospitals.

Ideal World Execution. The notion of security offered by GenSets is defined with respect to a relaxed variant of ideal world execution: upon receiving a query genome and a threshold t from the client and a list of genome strings from the database server, the trusted party generates a random string r and uses the approximation algorithm (described in Section 3.2) followed by applying a

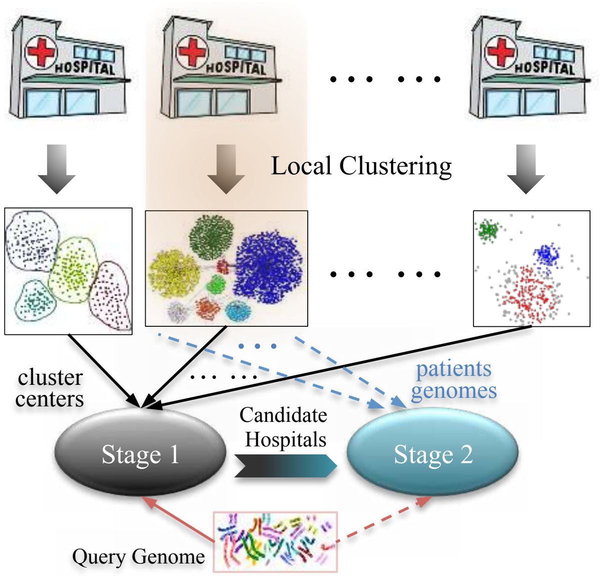

Figure 2: Two-stage secure SPQ with local clustering. Based on (weighted) edit distance metric, every hospital organizes their patients records in clusters. The first stage employs a secure SPQ primitive protocol to compare the querier's genome with every cluster center, in order to identify candidate hospitals that could have a similar patient. The second stage securely search through all patients in the candidate hospitals to identify all genetically similar individuals.

sketching algorithm (described in Section 3.4) with random tape r to compute all matching genomes in the server's list, and send to the client both the matches and the random tape r. This security notion is weaker, comparing to that developed around the standard ideal world execution where r is not revealed. However, we argue that, in practice, the additional leakage as result of revealing r is highly limited, though it remains open to formally prove this leakage is negligible.

3.2 Private Edit Distance Approximation

Our first key observation is that the actual input strings to the edit distance computation in the SPQ application are distributed in a very special way. For example, for any two un-related individuals, (1) much ($> 99.5\%$) of their DNA sequences are identical; (2) most ($> 95\%$) of their edits (from the reference genome) occur at non-adjacent locations; (3) most (about $80 \sim 90\%$) of the edits between their genomes are substitutions. We exploited these statistical features in designing an efficient (and also very accurate) approximation of edit distance between human genomes.

Secondly, we also observe that, assuming a public reference genome Ref, a significant portion of the computational task of private edit distance between any two human genomes can be moved into a pre-computable (and also amortizable) local preparation stage. Basically, each party pre-computes locally the minimum edits from Ref to their respective private genomes, and then launch a secure computation protocol to approximate edit distance just from the private edits.

Next, we present detailed approximation algorithms, followed by examples demonstrating why it works and when it does not.

The Algorithm. The protocol involves two parties, each having a private human genome as input. The whole approximation algorithm has three steps:

1. Each party calculates the minimum edit sequences from Ref to their own genomes. (In practice, edits of one genome (also known as variations) are stored in a VCF file.)
2. Each party computes a set of single-character edits from the minimum edit sequence associated with their private genome. Namely, every multi-character edit $e = (pos, op, aux)$ (where *pos* is the location of the edit, *op* is the type (either insert, delete, or substitute) of the edits, and *aux* represents operation-specific editing information) is decomposed into single-character edits as follows:

   Inserts: Inserting a string $\mathtt{c}_1 \ldots \mathtt{c}_\mathtt{n}$ at location *loc*, denoted as (*loc*, *ins*, $\mathtt{c}_1 \ldots \mathtt{c}_\mathtt{n}$), is translated into (*loc*, *ins*, 1, $\mathtt{c}_1$), (*loc*, *ins*, 2, $\mathtt{c}_2$), $\cdots$, (*loc*, *ins*, n, $\mathtt{c}_\mathtt{n}$).
   Deletes: Deleting a string of length n at location *loc*, denoted as (*loc*, *del*, *n*), is translated into (*loc*, *del*, 1), (*loc*+1, *del*, 1), $\cdots$, (*loc*+*n*-1, *del*, 1).
   Substitutes: Since substitutes are already defined with respect to a single character, no special treatment is needed to break them down.
3. The parties run a secure computation protocol to calculate the size of symmetric set difference between the two sets and output it as an approximation of the edit distance between the genomes. The symmetric set difference between sets A and B, denoted as $\mathit{Diff}\,(A, B)$, is defined as $(A - B) \cup (B - A)$.

Note that the first two steps only involve the public Ref and one party's genome, hence accomplishable with relatively inexpensive local computation. Moreover, they are also amortizable in the sense that they need to be done only once in a preparation stage, no matter how many queries are to be serviced. Only the third step requires more expensive secure computation, whose design is detailed in the next two subsections.

Examples. Suppose

Ref = ATTGCCCGA, A = GTTGGA, B = GTTCGA.

The minimum edits to convert Ref to A is $\{(1, sub, \mathtt{G}), (5, del, 3)\}$, and to convert Ref to B is $\{(1, sub, \mathtt{G}), (4, del, 3)\}$. Breaking down the edits into single-character edits, the two parties can respectively obtain set $A' = \{(1, sub, \mathtt{G}), (5, del, 1), (6, del, 1), (7, del, 1)\}$ and set $B' = \{(1, sub, \mathtt{G}), (4, del, 1), (5, del, 1), (6, del, 1)\}$. Therefore,$|\mathit{Diff}\,(A', B')| = 2$, which coincides with the edit distance between A and B.

Of course, there are cases our algorithm doesn't approximate very well. For instance, let

Ref = ATTGCCCGA, A = GTTGGATAA, B = GTTCGATGA.

In this case, the minimal sets of edits to obtain A and B from Ref are $\{(1, sub, \mathtt{G}), (4, ins, \mathtt{C}), (5, del, 1), (6, sub, \mathtt{A}), (7, sub, \mathtt{T})\}$ and $\{(1, sub, \mathtt{G}), (5, sub, \mathtt{G}), (6, sub, \mathtt{A}), (7, sub, \mathtt{T}), (8, sub, \mathtt{A})\}$, respectively. As A and B are already sets of single-character edits, it is obvious that $|\mathit{Diff}\,(A, B)| = 4$, whereas the actual edit distance between A and B is 2. The error is caused when comparing the 4th and 5th character of A and B— while CG can be converted to GG with just a single *sub* operation, the approximation algorithm essentially accounts it 3 times, namely, $(4, sub, \mathtt{C})$ and $(5, del, 1)$ for A and $(5, sub, \mathtt{G})$ for B, because they were derived from Ref in different ways.

Fortunately, these types of "problematic" scenarios happen very rarely in practice, because on human genomes, most (90% of) edits obtained in step 1 are short edits (involving $1 \sim 2$ nucleotides and the more problematic long-string, overlapping inserts and deletes almost never happen. In order to establish enough confidence over our approximation algorithm, we report below a comprehensive

Length of	Number of	Relative Error		
segments	tests	0.25%	0.5%	1%
80 million	$10,000$	78.15%	99.13%	100%

Figure 3: Accuracy of Approximation algorithm.

empirical study of comparing the end-results of our approximation with the ground truth distance values obtained from an edit distance implementation using dynamic programming, over genome snippets of various lengths.

Overall Accuracy. To understand the accuracy of this approximation, we computed edit distance between the genomes segments (each segment contains $8,000$ nucleotides) of two randomly selected individuals in dataset of the Personal Genome Project (PGP) [17] and compared the values with those produced by the estimation algorithm above. We note that because rigorous dynamic programming is not practical for the global alignment of genomic sequences with millions of bases, a common practice in genome comparison is to first find long identically matched segments in the input genomic sequences, and then to *chain* the aligned segment into global alignment [16, 13]. This is essentially approximating the global edit distance between two genome sequences by the sum of the edit distances between corresponding genome subsequences delineated by long identically matched segments. For comparing two human genomes with more than 99.5% identity, this *sum-of-segment* method should give the same edit distance as that computed by the rigorous dynamic programming algorithm. Therefore, in our experiment, we used the results of the sum-of-segment method as the ground truth distance for the comparison with our approximation. Still, it took 365 hours to compute the edit distance for $6,000$ tests cases.

In each test, we split long segments into shorter subsequences of varying lengths (i.e., eight sequences of $1,000$ nucleotides, four sequences of $2,000$ nucleotides, and two sequences of $4,000$ nucleotides) and then calculated the true edit distance using dynamic programming algorithm and approximation edit distance using our algorithm on segments and their subsequences. In our experiments, we observe that the edit distance between two $8,000$-nucleotide sequences is always exactly the same as the sum of the two edit distance values over its two $4,000$-nucleotide components; and same observation applies to segments of length $4,000$ and $2,000$ as well. These results suggest that, over real world human genome data, edit distance between long sequences can be accurately computed from sum of distances on its (not too short) subsequences.

Based on the observation above, we studied the accuracy of our approximation algorithm on longer sequences (each contains 80 million nucleotides), by summing up distance values over $10,000$ random basic segments (each contains $8,000$ nucleotides). Our experimental results show that 99.13% of $10,000$ tests exhibited an error rate less than 0.5%, while all tests resulted in less than 1% error (Figure 3). These results demonstrate the accuracy of our approximation algorithm on real human genome data.

3.3 Private Set Difference Size

Our approximation algorithm above reduces a private human genome edit distance problem to a private set difference size problem. Next, we will describe a basic protocol for *private set difference size*, and present in Section 3.4 three variations of the basic protocol each best suited for certain scenarios.

Problem Definition. Given two secret sets A and B, output $|\mathit{Diff}(A,B)|$ without revealing anything else about A and B. In particular, in the context of our private edit distance approximation scheme, we hope to efficiently handle sets of $5 \sim 10$ million elements, with accuracy comparable to that of the original approximation algorithm discussed above.

Some Strawman Solutions. Since $|\mathit{Diff}(A,B)| = |A| + |B| - 2 \cdot |A \cap B|$ where $|A|$ and $|B|$ (the sizes of A and B) are not secret, $|\mathit{Diff}(A,B)|$ could be derived from $|A \cap B|$. One may attempt to first securely compute the set intersection of A and B using a Bloom Filter [20, 14] then securely count the number of elements in the intersection and infer the size of set difference. One could even leverage a Bloom Filter cardinality estimator [48] to save the expensive secure counting phase. Alternatively, one may even construct a secure version of the Count-Min sketch [18] or HyperLoglog [25] to directly estimate the intersection size without using a Bloom Filter. However, these solutions do not work in practice for SPQ because the error introduced in estimating the intersection size needs be multiplied $|A \cap B| / \mathit{Diff}(A,B)$ times when it comes to describing the relative error in estimating set difference size. In SPQ, the most interesting data points (which indicates matching patients) actually fall into the category where $|A \cap B| / \mathit{Diff}(A,B)$ is very large. Tuning up the parameters (which grow at a rate of $O(1/\varepsilon)$ to $O(1/\varepsilon^2)$ where ε is the error rate) of these intersection size estimators to achieve reasonably small error relative to set difference size will degrade SPQ performance to an unusable level.

A Basic Solution. We propose highly efficient and accurate protocols that directly compute set difference size. For the purpose of illustration, we first present a basic solution (Figure 4), from which the more efficient variants are later derived.

Inspired by the seminal work by Alon et al. [2] and Feigenbaum et al. [24], the basic idea is to first "compress" every input set, say S, into a single integer d_S, using a binary hash function $h : U \mapsto \{-1, 1\}$, where U denotes the universe of all set elements. More concretely, d_S is defined to be $\sum_{s \in S} h(s)$. Assuming h can be randomly sampled from a family of pairwise independent binary hash functions, so for any element s, s_1, s_2 ($s_1 \neq s_2$) and a randomly sampled h, it is easy to see that $E[h(s)] = 0$, $E[h^2(s)] = 1$, and $E[h(s_1)h(s_2)] = E[h(s_1)] \cdot E[h(s_2)] = 0$, with all probabilities taken over the randomness in sampling h. Thus, for any set S,

$$\begin{aligned} E\left[d_S^2\right] &= E\left[\left(\sum_{s \in S} h(s)\right)^2\right] \\ &= E\left[\sum_{s \in S} h^2(s) + 2 \cdot \sum_{s_1 \neq s_2} h(s_1)h(s_2)\right] \\ &= E\left[\sum_{s \in S} h^2(s)\right] = |S|. \end{aligned}$$

Let d_A, d_B be the sketch integers computed from the two private input sets A and B, respectively. If we further assume the family of hash functions are four-wise independent (namely, for all distinct s_1, s_2, s_3, s_4, $E[h(s_1)h(s_2)h(s_3)h(s_4)] = E[h(s_1)]E[h(s_2)]E[h(s_3)]E[h(s_4)]$), then we can show that $E\left[(d_A - d_B)^2\right] = |\mathit{Diff}(A,B)|$. This is because

$$d_A - d_B = \sum_{s \in A} h(s) - \sum_{s \in B} h(s) = \sum_{s \in A - B} h(s) - \sum_{s \in B - A} h(s).$$

```
The Basic Protocol

Input of A: a set $A$.
Input of B: a set $B$.
Public Input: a (sufficiently long) common random string
Output: $|Diff(A, B)|$
1. For $j$ from 1 to $k$
    (a) For $i$ from 1 to $\ell$
        i. A and B use the (same) random string to randomly pick
           a function $h$ from the family of hash functions.
        ii. A computes $d_A = \sum_{s \in A} h(s)$, whereas B computes
            $d_B = \sum_{s \in B} h(s)$, independently.
        iii. A and B run a secure computation protocol with re-
             spective private inputs $d_A$ and $d_B$, to compute $D_i = (d_A - d_B)^2$.
    (b) A and B securely compute $\hat{D}_j = \sum_i^{\ell} D_i/\ell$.
2. A and B securely compute the median $Z$ of $\hat{D}_1, \cdots, \hat{D}_k$.
   Output $Z$.
```

Figure 4: The basic protocol to approximate set difference size.

Therefore,

$$\begin{aligned} E\left[(d_A - d_B)^2\right] = E\Bigg[& \left(\sum_{s \in A-B} h(s)\right)^2 + \left(\sum_{s \in B-A} h(s)\right)^2 \\ & +2 \cdot \left(\sum_{s_1 \in A-B} h(s_1)\right) \cdot \left(\sum_{s_2 \in B-A} h(s_2)\right)\Bigg] \\ = {} & |A - B| + |B - A| + 2 \cdot 0 = |Diff(A, B)|. \end{aligned}$$

To efficiently bound the error, our basic protocol estimates $Diff(A, B)$ by computing the k-median of ℓ-mean of random sampling of $(d_A - d_B)^2$.

THEOREM 0. *For any two sets A, B, let $d = |Diff(A, B)|$ and X be the output of the Basic Protocol running with A, B. Then for any positive ε and any positive integer λ, the inequality*

$$Pr\{|X - d| \geq \varepsilon d\} \leq 2^{-\lambda}$$

can be achieved by setting $\ell = O(1/\varepsilon^2), k = O(\lambda)$.

PROOF. See the proof in Appendix A. ▮

Cost Analysis. In this basic protocol, Step 1(a)i and Step 1(a)ii can be done locally without expensive secure computation. Both steps are executed $k\ell$ times, while the cost of Step 1(a)ii also grows linearly with the size of the set. Note that for a whole-genome, the size of set A (or B) will be $5 \sim 10$ million. Also taking the factor $k\ell$ (whose exact value depends on the accuracy needed) into account, the cost of the local hashing step can be substantial. In Section 3.4, we give several techniques to reduce this cost.

Expensive secure computation is required only at three places (i.e., the Step 1(a)iii, 1b and 2), of which the Step 1(a)iii dominates the cost. If d_A and d_B are ω-bit integers, then overall, it incurs $k\ell$ ω-bit subtractions, $k\ell$ ω-bit integer squaring, ℓ 2ω-bit additions, and one secure median of k 2ω-bit integers. Thus, while the dominant cost comes from secure squaring, the overall cost of secure computation also highly depends on the integer bit length ω.

Hashing. A 4-wise universal hash function can be generated by picking a random polynomial modulo a large prime. This, however, requires many multiplication operations per element. Therefore, we customized `murmurHash64` (by leaving out several instructions that don't affect the accuracy of our approximation algorithm) and used it in our prototype. In addition, our implementation fully utilizes all 64 bits of the hash result to compute 64 $d_A(d_B)$s at the same time.

```
Protocol 1

Input of A: a set $A$.
Input of B: a set $B$.
Public Input: a (sufficiently long) common random string
Output: $|Diff(A, B)|$.
1. For $j$ from 1 to $k$
    (a) A and B use the public common random string to ran-
        domly sample a hash function $h : U \mapsto \{-1, 1\}$ and a
        hash function $g : U \mapsto \{1, 2, \cdots, \ell\}$.
    (b) A and B initialize arrays $d_A$ and $d_B$ (each of length $\ell$,
        respectively, to all zeros.
    (c) A computes $d_A[g(s)] := d_A[g(s)] + h(s)$ for every
        $s \in A$; while B computes $d_B[g(s)] := d_B[g(s)] + h(s)$
        for every $s \in B$.
    (d) A and B run a secure computation protocol to securely
        compute $D_j = \sum_{i=1}^{\ell} (d_A[i] - d_B[i])^2$
2. A and B use a secure computation protocol to securely com-
   pute the median $Z$ of $D_1, \cdots, D_k$. Output $Z$.
```

Figure 5: Faster private set difference size through bucketing. U denotes the universe of all set elements.

Oblivious Transfer. When this protocol is actually deployed between hospital servers and health professional querier clients, the querier provides only one genome sketch per query while a hospital would need to provide thousands of private genome sketches per query. We take advantage of this asymmetry by setting up the hospital to be the garbled circuit generator so that only the querier's short inputs needs to be oblivious transferred in each query.

3.4 Optimized Protocols

Although our basic protocol above already outperforms the strawman solutions, there is ample design space to explore to further improve efficiency and accuracy. Next, we present a few interesting optimizations that aim to reduce the cost of local hashing and secure computation, while retaining accuracy.

3.4.1 Protocol 1

Inspired by the work of Thorup and Zhang on tabulating hashes in second moment estimation [50], we improve the basic private set difference size protocol through random *bucketing*. The full protocol is given in Figure 5. The basic idea is to require each party to associate every set element s with one single (out of ℓ in total) bucket, according to a hash of s, namely $g(s)$ in Figure 5; and then estimate the number of elements in $Diff(A, B)$ falling in each bucket simply with $(d_A[i] - d_B[i])^2$, and finally sum the numbers up. Like in the basic protocol, the median function is also used to bound the estimation error. Now it involves only a single loop of k iterations, and in each iteration an element will be hashed only once. The key observation is that although an ℓ-times coarser estimator is used to measure the size of each bucket (hence saving a factor of ℓ hashes), the variance of $(d_A[i] - d_B[i])^2$ is actually ℓ-times smaller thanks to bucketing. Thus, it achieves the same level of accuracy with ℓ-times less hashing compared to the basic protocol.

THEOREM 1. *For any two sets A, B, let $d = |Diff(A, B)|$ and X be the output of Protocol 1 running with A, B. Then for any*

```
Protocol 2

Input of A: a set $A$.
Input of B: a set $B$.
Public Input: a (sufficiently long) common random string
Output: $|Diff(A, B)|$.
1. For $j$ from 1 to $k$
   (a) For $i$ from 1 to $\ell$
       i. A and B use the (same) random string to randomly pick
          a function $h$ from the family of hash functions.
       ii. A computes $d_A = \sum_{s \in A} h(s)$, whereas B computes
           $d_B = \sum_{s \in B} h(s)$, independently.
       iii. Party A and B run a secure computation protocol
            with inputs $d_A$ and $d_B$, respectively, to compute
            $\underline{\underline{D_i = \sqrt{\pi/2} \cdot |d_A - d_B|}}$.
   (b) A and B securely compute $\hat{D}_j = \sum_i^{\ell} D_i/\ell$.
2. A and B securely compute the median $Z$ of $\hat{D}_1, \cdots, \hat{D}_k$.
   $\underline{\underline{\text{Output } Z^2}}$.
```

Figure 6: Faster set difference size without secure squaring.

positive ε and any positive integer λ, the inequality

$$\Pr\{|X - d| \leq \varepsilon d\} \geq 2^{-\lambda}$$

can be achieved by setting $\ell = O(1/\varepsilon^2), k = O(\lambda)$.

PROOF. See the proof in Appendix C. ▮

Remark. The benefits of randomized bucketing actually go beyond reducing the number of local hashes. Since the number of times the accumulators $d_A[i]$ and $d_B[i]$ are incremented is reduced by ℓ times, the number of bits (ω) in $d_A[i], d_B[i]$ is then reduced by $\log \ell$, which saves substantial cost ($30 \sim 40\%$) in the secure computation stage.

3.4.2 Protocol 2

As is mentioned earlier, secure squaring accounts for the dominant cost of the secure computation part of the protocol. Fortunately, under an extra assumption that $(d_A - d_B)$ is very close to normal distribution, it can be shown that $E(|d_A - d_B|) = \sigma\sqrt{2/\pi}$ (where $\sigma^2 = E\left[(d_A - d_B)^2\right] - (E[|d_A - d_B|])^2$) because $|d_A - d_B|$ is a *half-normal* distribution [3]. Thus, measuring $E(|d_A - d_B|)$ suffices to provide a good estimation of $E\left[(d_A - d_B)^2\right]$. Because $d_A - d_B$ is a binomial distribution, it is indeed very close to normal distribution as the set difference sizes observed in our genomic SPQ application all turn out much larger than 10,000.

Figure 6 describes the improved protocol based on this observation (and the extra assumption that the size of set difference is not too small). The protocol resembles that of the basic protocol except for two changes highlighted with double-underlines in Step 1(a)iii and Step 2.

THEOREM 2. *For any two sets A, B, let $d = |Diff(A, B)|$ and X be the output of Protocol 2 running with A, B. Then for any positive ε and any positive integer λ, the inequality*

$$Pr\{|X - d| \leq \varepsilon d\} \geq 2^{-\lambda}$$

can be achieved by setting $\ell = O(1/\varepsilon^2), k = O(\lambda)$.

PROOF. See the proof in Appendix C. ▮

We remark that it is unclear whether it is possible to harvest the savings by combining both ideas in one protocol, as the two optimizations are not compatible when trivially combined. To see the reason, imagine a third protocol randomly places set elements in ℓ buckets and attempts to use non-squaring approach to estimate the number of difference elements in each bucket. Because the number of elements in each bucket needs to be kept secret, the squaring operation of Step 2 in Protocol 2 needs to be done securely. Note that this per bucket squaring can't be simply moved to the end because for $E\left[\sum_{i=1}^{k} {Z_i}^2\right]$ (where Z_i is obtained for the i-th bucket) to be equal to $Diff(A, B)$, Z_i's need to be at least pairwise independent. However, this is not the case as each element was hashed into one and only one (out of ℓ) bucket.

3.4.3 Thresholding private set difference size

Although the protocols above are able to securely approximate the size of set difference with arbitrary precision, the key primitive that fits best with a secure SPQ (such as *PatientLikeMe*) is actually comparing the set difference size with a given threshold, producing merely 1-bit output. In fact, thresholding the difference size is more desirable because it limits the potential information leak through output. A private set difference size protocol can be trivially extended to provide the secure thresholding primitive. However, it is worth noting that secure thresholding protocols generally result in much less error compared to the corresponding private set difference size protocols from which they are derived, due to a difference in the notion of error: 1% error in the size of set difference implies the estimated value is 1% away from the true value; while 1% error in thresholding the size actually means the chance to arrive at a wrong decision is 1%. As a result, smaller parameters (k, ℓ values) suffice to achieve the same level of accuracy. We have done extensive experiments to evaluate the performance and accuracy of secure thresholding protocols (see Section 4.1).

3.5 SPQ Infrastructure

Based on the secure SPQ primitive described above, we implemented a prototype infrastructure to support a secure SPQ over a large amount of data across multiple institutions. Such an infrastructure could be improved to work in a practical scenario where hundreds of thousands of genomes (associated with a particular disease, e.g., the breast cancer) collected by hundreds of hospitals are scanned to find those genetically similar to the patient in the query. To make this operation efficient at large scales, we design a data pre-processing mechanism for each hospital to organize its patient data into clusters.

Clustering. We leverage the complete-linkage hierarchical clustering algorithm [34] (other more sophisticated algorithms can also be plugged into our infrastructure) to cluster patient genomes based on our edit distance approximation (Section 3.2). Note that the notion of clusters here does not necessarily match up with a known pathological categorization. In each cluster, a synthesized sequence, called *representative*, is generated. Let δ be the radius of each cluster, and ϵ be the threshold used to identify similar patients. Due to the triangular property of edit distance, if a patient P in a cluster C is considered similar to the querier's patient Q, then the distance between Q and the representative of C should be no more than $\delta + \varepsilon$.

Two-stage search. With these pre-computed clusters, a SPQ search will be performed in two stages (Figure 2):

1. Compare the edit distance between the query genome and each cluster representative with threshold $\delta + \varepsilon$ using a se-

cure thresholding scheme: if the distance is below the threshold, the *hospital* owning this cluster will be selected for the second stage search.

2. Compare the edit distance between the query genome and each patient in every selected hospital with threshold ε using a secure thresholding scheme: if the distance is below the threshold, the pseudo-identifier of the patient will be returned.

In the particular case of SPQ, ε is chosen to be much smaller than the edit distance between two random patients within the same cluster, i.e., $\varepsilon \ll \delta$. This allows us to use δ as the threshold for the first stage search to efficiently eliminate clusters that contain no patients similar to the query. Note that for a query involving a similar patient a in cluster A, the distance between the query genome and the representative of any other cluster is unlikely below $\delta + \varepsilon$ (as $\varepsilon \ll \delta$) unless the query genome (and its similar patient) is close to the border of the cluster.

4. EVALUATION

In this section, we present experimental evaluation of both accuracy and efficiency of GENSETS.

System and network. Unless explicitly specified otherwise, all experiments were performed between two machines located more than 2000 miles apart (one in Bloomington, Indiana and the other in San Diego, California). The bandwidth is about 100Mbps with variations. We run the garbled circuit and oblivious transfer protocols using a single thread. We exploited multi-core parallelism to compute the amortizable precomputed hashing phase of sketch construction.

The implementation of the garbled circuit protocol leverages half-gate garbling [55] and free-XOR technique [36]. The oblivious transfer is implemented using NPOT [42] with OT extension [32] of Ishai et al.

Metrics. In our evaluation, we use the following metrics to measure the accuracy and efficiency of our approach.

1. **False positive/negative rate.** A *false positive* refers to the event that a patient dissimilar to the query is returned; while a *false negative* happens when a similar patient is not returned.
2. **Error rate.** Error rate measures the accuracy of private set difference size protocols. It is defined to be $|u-v|/u$ where u denotes the true size and v is what the secure protocol outputs.
3. **Number of AND gates.** With garbled circuit protocol, the main cost grows linearly with the number of AND gates in the circuit.
4. **Wall-clock time.** This is the total elapsed time of a task.

4.1 Thresholding Private Set Difference Size

Private set difference size thresholding protocols are the core SPQ primitive that enables GENSETS. Figure 7 and Figure 8 show the performance of our thresholding protocols running over breast cancer patients' genomes (each of which is represented by a VCF file of roughly 150K variations). The private edit distance protocol is reduced to thresholding the set difference size of two sets, each containing $200K \sim 300K$ single-character edits.

We observe that the accuracy achieved is generally proportional to the value of kl (at least when the difference in k is small). Also note the asymmetry of errors in the range around the threshold. For instance, comparing the columns $d = 0.9t$ and $d = 1.1t$, the false negative rates (on columns with $d < t$) is always smaller than the false positive rates (on columns with $d > t$) on their mirroring columns. This would be desirable in SPQ search as the users are usually more sensitive on false negatives (i.e., similar patients are overlooked) while tending to tolerate false positives (irrelevant patients are returned). Last, note that protocol 1 and 2 performs comparably in thresholding the set difference size, except that protocol 2 is about 30% faster, while protocol 1 is slightly better in terms of accuracy.

In our particular application of secure SPQ, we assume there are 250 hospitals, each of which keeps 4000 patients records organized in 8 clusters. The first stage would check a total of $250 \times 8 = 2000$ clusters. Assuming at most 5 hospitals will be selected as candidates to proceed in the second stage search, which amounts to searching through all $4000 \times 5 = 20000$ patients in these 5 hospitals. Since cluster centers have the same representation as patient genomes, the performance of the entire search for similar breast cancer patients is equivalent to checking 22000 patients. Using protocol 2 with parameter $k = 5, \ell = 512$, 22000 edit-distance comparison can be accomplished within 183 minutes.

4.2 Private Set Difference Size

In an SPQ scenario, calculating the set difference size with high precision is mostly unnecessary. However, once a similar patient is found, it may be worthwhile to calculate the edit distance with high accuracy to confirm the match. Moreover, many other personal genomic applications (such as genetic diagnosis and medical treatment risk prediction) may find it useful to be able to precisely estimate edit distance in a privacy-preserving way.

Figure 9 shows the performance of Protocol 1 and Protocol 2 used in private set difference size estimation scenario. For each protocol, we report the cost to bound 90%-percentile error rate to 1% and 0.5%, respectively. This means that for 90% of the test cases, the relative error is less than or equal to 1% or 0.5% respectively. The results show that the protocols runs significantly slower than private set difference size thresholding protocols. However, since the number of similar patients returned in the final stage is usually quite low (typically less than 10), users can afford much more time per candidate patients for obtaining an accurate distance.

Note that, thanks to the optimized configuration of the OT protocol (Section 3.3), the cost of OT is independent of number of patients on the hospital server. Therefore, the total running time to query n patients can be calculated as

$$\text{Time}_{\text{Total}} = \text{Time}_{\text{OT}} + n \times (\text{Time}_{\text{Local}} + \text{Time}_{\text{GC}})\,.$$

It is easy to verify that the total times reported in the figure conforms to the formula above with $n = 10$.

4.3 Experiments on Whole Genomes

We have also measured the performance of our protocols over whole-genomes obtained from the PGP project. Figure 10 shows the total running time of our approach on whole genomes. First we find that the timings for whole genome data are about $4 \sim 5$ times slower than those of breast cancer tests, because the genomes considered in breast cancer tests are only a fraction $(1/40)$ of whole genomes. The primary cause of the slowdown is the increased cost of garbled circuit generation and evaluation, since the bit length ω of the sums of the hashes is increased by a factor of around $\log 40 \approx 5.3$). Secondarily, the local hashing and oblivious transfer, whose costs grow linear with the length of the input genomes, are 40 times more expensive.

5. RELATED WORK

Secure Approximation. Feigenbaum et al. [23] first considered the problem of secure approximation. Generally a streaming algorithm consists of phases to locally compute the sketches and those

k	ℓ	Set Difference Siz (d)								1000 Patients	
		$0.7t$	$0.8t$	$0.9t$	$0.95t$	$1.05t$	$1.1t$	$1.2t$	$1.3t$	Running time	Bandwidth
		False negative rate				False positive rate					
3	32	0.01%	0.06%	0.22%	0.33%	0.43%	0.33%	0.17%	0.09%	47.13s	0.17GB
	64	0.0%	0.02%	0.15%	0.3%	0.37%	0.24%	0.08%	0.02%	59.84s	0.34GB
	128	0.0%	0.0%	0.09%	0.24%	0.31%	0.15%	0.02%	0.0%	92.96s	0.67GB
	256	0.0%	0.0%	0.03%	0.18%	0.22%	0.06%	0.0%	0.0%	165.57s	1.35GB
	512	0.0%	0.0%	0.01%	0.1%	0.12%	0.01%	0.0%	0.0%	304.16s	2.69GB
5	32	0.0%	0.03%	0.17%	0.29%	0.42%	0.29%	0.12%	0.04%	54.78s	0.29GB
	64	0.0%	0.01%	0.1%	0.25%	0.35%	0.19%	0.04%	0.0%	80.25s	0.57GB
	128	0.0%	0.0%	0.05%	0.2%	0.25%	0.09%	0.01%	0.0%	173.34s	1.13GB
	256	0.0%	0.0%	0.01%	0.12%	0.16%	0.03%	0.0%	0.0%	244.81s	2.25GB
	512	0.0%	0.0%	0.0%	0.05%	0.08%	0.0%	0.0%	0.0%	596.79s	4.49GB

Figure 7: Thresholding set difference size using extended protocol 1. (using breast cancer patients' genome) When $k = 3, \ell = 256$, given a threshold t, this algorithm achieves a false negative rate of 0.03% if the set difference size is $0.9t$; and achieves a false positive rate of 0.06% if the set difference size is $1.1t$.

k	ℓ	Set Difference Size (d)								1000 Patients	
		$0.7t$	$0.8t$	$0.9t$	$0.95t$	$1.05t$	$1.1t$	$1.2t$	$1.3t$	Running time	Bandwidth
		False negative rate				False positive rate					
3	32	0.02%	0.09%	0.26%	0.38%	0.41%	0.32%	0.17%	0.08%	31.7s	0.11GB
	64	0.0%	0.03%	0.19%	0.33%	0.36%	0.24%	0.09%	0.02%	44.05s	0.22GB
	128	0.0%	0.01%	0.12%	0.27%	0.3%	0.16%	0.02%	0.0%	66.26s	0.44GB
	256	0.0%	0.0%	0.04%	0.2%	0.23%	0.07%	0.0%	0.0%	131.98s	0.96GB
	512	0.0%	0.0%	0.01%	0.12%	0.14%	0.02%	0.0%	0.0%	226.2s	1.75GB
5	32	0.0%	0.05%	0.21%	0.34%	0.38%	0.27%	0.11%	0.04%	43.06s	0.21GB
	64	0.0%	0.01%	0.13%	0.28%	0.33%	0.19%	0.04%	0.01%	62.83s	0.37GB
	128	0.0%	0.0%	0.06%	0.21%	0.27%	0.1%	0.01%	0.0%	99.89s	0.73GB
	256	0.0%	0.0%	0.02%	0.15%	0.17%	0.03%	0.0%	0.0%	224.96s	1.6GB
	512	0.0%	0.0%	0.0%	0.07%	0.09%	0.0%	0.0%	0.0%	497.51s	3.2GB

Figure 8: Thresholding set difference size using extended Protocol 2. (using breast cancer patients' genome)

	k	ℓ	90%	Total time /10 Patients	Total time /100 Patients
Protocol 1	5	8192	1.42%	390s	3970s
Protocol 2			1.42%	401s	4014s

Figure 10: Running Protocol 1 and 2 on whole genomes.

to jointly combine the sketches. Feigenbaum et al. pointed out that it is unclear how to prove it secure if secure computation protocol is not used to compute the sketches, (although no actual attacks are identified). We get around this issue by relaxing the security definition to allow revealing the randomness used in the sketch computation in the ideal world execution.

Privacy-preserving Genome Analysis. Researchers proposed solutions to privately compare two genomes, either using a private set intersection (PSI) protocol [6], or a private set intersection size protocol [15]. These works, however, modeled the similarity by Hamming distance between genomes and could only handle relatively short genome snippets. In contrast, our work targets at the more challenging (and also generally more useful) edit distance protocol applied to people's *whole-genomes*. In addition, the private set difference size protocol runs *many orders of magnitude faster* than those derived from the state-of-the-art PSI or PSI size protocols.

Streaming Algorithms. Streaming algorithms, introduced by the groundbreaking work of Alon et al. [2], aim to significantly improve the (space, communication, and time) efficiency of problem solutions by tolerating a (controlled) small error. We borrowed many useful ideas from streaming algorithms research in designing the private set difference size protocols. However, different from the existing research on streaming algorithm, we studied the problem in secure computation setting, thus exploring the design space under a different set of goals, e.g., efficient joint computation versus small working space and low plaintext communication.

Approximating Edit Distance Quite a few researchers have considered approximating edit distance for general input strings, particularly in the streaming setting [4, 7]. However, the best algorithms still incur significant error (such as a factor of $\sqrt{n}$ where n is the length of the input string), which renders them inapplicable in practice. In addition, none of the existing work considers the cost of executing the sketch combining phase with secure computation protocols.

Approximating Set Difference Size. Feigenbaum et al. [24] proposed an efficient algorithm to compute L-1 norm. But their work considers the streaming setting and is general enough to compute weighted L-1 norm over multi-sets. Our basic protocol in Section 3.3 was inspired by their work, but is highly customized towards solving the privacy-preserving SPQ problem. For example,

	k	ℓ	90%	Oblivious Transfer	Local Hash / patient	GC time / patient	#AND gates / patient	Bandwidth / patient	Total time / 10 patients
Protocol 1	5	8192	1.4%	$5.83s$	$0.01s$	$9.43s$	3851.0K	73.44MB	$95.1s$
Protocol 1	5	16384	1.0%	$9.18s$	$0.01s$	$18.98s$	7701.0K	146.88MB	$199.08s$
Protocol 1	5	65535	0.5%	$42.06s$	$0.01s$	$71.4s$	27.8M	531MB	$756s$
Protocol 2	5	8192	1.4%	$13.05s$	$0.37s$	$7.18s$	2745.0K	52.35MB	$76.84s$
Protocol 2	5	16384	1.0%	$29.8s$	$0.92s$	$15.71s$	5489.0K	104.7MB	$196.1s$
Protocol 2	5	65535	0.5%	$125.28s$	$2.83s$	$56.52s$	22M	419MB	$718.78s$

Figure 9: Private set difference size using Protocol 1 and Protocol 2. (using breast cancer patients' genomes) The cost of oblivious transfer is per query, as it is setup so that their costs are independent of the number of patients in server's dataset (see Section 3.3).

we avoided algebraic computations over finite fields required in their construction. Several other works [27, 43] have further improved the update time of this scheme, but still only considering the classic streaming model.

Using invertible bloom filter based approximation, Eppstein et al. [21] proposed protocols to reconstruct set difference and estimate set difference size. Although their set difference size approximation algorithm achieved a similar asymptotic error bound (in terms of sketch size), their algorithm is more expensive in the secure computation model (because their construction focused on minimizing the size of the sketch while ours aims at reducing the cost of joint computation).

6. CONCLUSION

Securely computing edit distance between human whole-genomes promises many interesting applications of personal genomic data in medical and public health domains. We described novel techniques that is able to approximate edit distance on human genomes with unprecedented efficiency and accuracy. Based on the primitives we proposed, we implemented and evaluated GenSets, a genome-wide secure SPQ system. The performance of SPQ demonstrated in our experiments with realistic genomic data and network setting shows that we have made a big step towards privacy-preserving SPQ at the national scale.

7. ACKNOWLEDGEMEMNT

We would like to thank Jonathan Katz for numerous helpful discussions. This work is supported by NSF award #1111599, #1464113, #1117106, #1223477, #1223495, #1408874 and NIH HG007078.

8. REFERENCES

[1] PatientsLikeMe. https://www.patientslikeme.com. Accessed on May 8, 2015.

[2] Alon, N., Matias, Y., and Szegedy, M. The space complexity of approximating the frequency moments. In *STOC* (1996).

[3] Altman, D. G. Construction of age-related reference centiles using absolute residuals. *Statistics in medicine* (1993).

[4] Andoni, A., and Onak, K. Approximating edit distance in near-linear time. In *41st STOC* (2009).

[5] Atallah, M. J., Kerschbaum, F., and Du, W. Secure and private sequence comparisons. In *Proceedings of the 2003 ACM workshop on Privacy in the electronic society* (2003).

[6] Baldi, P., Baronio, R., De Cristofaro, E., Gasti, P., and Tsudik, G. Countering gattaca: efficient and secure testing of fully-sequenced human genomes. In *CCS* (2011).

[7] Bar-Yossef, Z., Jayram, T. S., Krauthgamer, R., and Kumar, R. Approximating edit distance efficiently. In *45th FOCS* (2004).

[8] Beaver, D. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology* (1991).

[9] Beaver, D. Correlated pseudorandomness and the complexity of private computations. In *STOC* (1996).

[10] Beaver, D. Commodity-based cryptography (extended abstract). In *STOC* (1997).

[11] Bellare, M., Hoang, V. T., Keelveedhi, S., and Rogaway, P. Efficient garbling from a fixed-key blockcipher. In *IEEE S & P* (2013).

[12] Ben-Or, M., Goldwasser, S., and Wigderson, A. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC* (1988).

[13] Blanchette, M., Kent, W. J., Riemer, C., Elnitski, L., Smit, A. F., Roskin, K. M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E. D., et al. Aligning multiple genomic sequences with the threaded blockset aligner. *Genome research* (2004).

[14] Bloom, B. H. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* (1970).

[15] Blundo, C., De Cristofaro, E., and Gasti, P. Espresso: efficient privacy-preserving evaluation of sample set similarity. *Journal of Computer Security* (2014).

[16] Brudno, M., Do, C. B., Cooper, G. M., Kim, M. F., Davydov, E., Green, E. D., Sidow, A., Batzoglou, S., Program, N. C. S., et al. Lagan and multi-lagan: efficient tools for large-scale multiple alignment of genomic dna. *Genome research* (2003).

[17] Church, G. M. The personal genome project. *Molecular Systems Biology* (2005).

[18] Cormode, G., and Muthukrishnan, S. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms* (2005).

[19] Damgård, I., Pastro, V., Smart, N. P., and Zakarias, S. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO* (2012).

[20] Dong, C., Chen, L., and Wen, Z. When private set intersection meets big data: an efficient and scalable protocol. In *CCS* (2013).

[21] Eppstein, D., Goodrich, M. T., Uyeda, F., and Varghese, G. What's the difference?: efficient set

reconciliation without prior context. In *ACM SIGCOMM Computer Communication Review* (2011).

[22] EVANS, W. E., AND RELLING, M. V. Moving towards individualized medicine with pharmacogenomics. *Nature* (2004).

[23] FEIGENBAUM, J., ISHAI, Y., MALKIN, T., NISSIM, K., STRAUSS, M. J., AND WRIGHT, R. N. Secure multiparty computation of approximations. In *Automata, Languages and Programming*. 2001.

[24] FEIGENBAUM, J., KANNAN, S., STRAUSS, M., AND VISWANATHAN, M. An approximate l1-difference algorithm for massive data streams. *SIAM Journal of Computing* (2002).

[25] FLAJOLET, P., FUSY, É., GANDOUET, O., AND MEUNIER, F. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *DMTCS Proceedings* (2008).

[26] FREDERIKSEN, T. K., JAKOBSEN, T. P., NIELSEN, J. B., NORDHOLT, P. S., AND ORLANDI, C. MiniLEGO: Efficient secure two-party computation from general assumptions. In *EUROCRYPT* (2013).

[27] GANGULY, S., AND CORMODE, G. On estimating frequency moments of data streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. 2007.

[28] GOLDREICH, O., MICALI, S., AND WIGDERSON, A. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC* (1987).

[29] HEENEY, C., HAWKINS, N., DE VRIES, J., BODDINGTON, P., AND KAYE, J. Assessing the privacy risks of data sharing in genomics. *Public health genomics* (2011).

[30] HUANG, Y., EVANS, D., KATZ, J., AND MALKA, L. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium* (2011).

[31] HUANG, Y., KATZ, J., AND EVANS, D. Quid-Pro-Quo-tocols: Strengthening semi-honest protocols with dual execution. In *IEEE S & P* (2012).

[32] ISHAI, Y., KILIAN, J., NISSIM, K., AND PETRANK, E. Extending oblivious transfers efficiently. In *CRYPTO 2003*.

[33] ISHAI, Y., PRABHAKARAN, M., AND SAHAI, A. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO* (2008).

[34] JAIN, A. K., DUBES, R. C., ET AL. *Algorithms for clustering data*. 1988.

[35] JARECKI, S., AND SHMATIKOV, V. Efficient two-party secure computation on committed inputs. In *EUROCRYPT* (2007).

[36] KOLESNIKOV, V., AND SCHNEIDER, T. Improved garbled circuit: Free XOR gates and applications. In *ICALP* (2008).

[37] LINDELL, Y., AND PINKAS, B. Privacy preserving data mining. In *CRYPTO* (2000).

[38] LINDELL, Y., AND PINKAS, B. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *EUROCRYPT* (2007).

[39] LINDELL, Y., AND PINKAS, B. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology* (2009).

[40] LIU, C., HUANG, Y., SHI, E., KATZ, J., AND HICKS, M. W. Automating efficient RAM-model secure computation. In *IEEE S & P* (2014).

[41] MALKHI, D., NISAN, N., PINKAS, B., AND SELLA, Y. Fairplay — a secure two-party computation system. In *USENIX Security Symposium* (2004).

[42] NAOR, M., AND PINKAS, B. Efficient oblivious transfer protocols. In *SODA* (2001).

[43] NELSON, J., AND WOODRUFF, D. P. Fast manhattan sketches in data streams. In *PODS* (2010).

[44] NETWORK, C. G. A., ET AL. Comprehensive molecular portraits of human breast tumours. *Nature* (2012).

[45] NIELSEN, J. B., NORDHOLT, P. S., ORLANDI, C., AND BURRA, S. S. A new approach to practical active-secure two-party computation. In *CRYPTO* (2012).

[46] PINKAS, B., SCHNEIDER, T., SMART, N. P., AND WILLIAMS, S. C. Secure two-party computation is practical. In *ASIACRYPT* (2009).

[47] SHERRY, S. T., WARD, M.-H., KHOLODOV, M., BAKER, J., PHAN, L., SMIGIELSKI, E. M., AND SIROTKIN, K. dbsnp: the ncbi database of genetic variation. *Nucleic acids research* (2001).

[48] SWAMIDASS, S. J., AND BALDI, P. Mathematical correction for fingerprint similarity measures to improve chemical retrieval. *Journal of chemical information and modeling* (2007).

[49] TAYLOR, J. G., CHOI, E.-H., FOSTER, C. B., AND CHANOCK, S. J. Using genetic variation to study human disease. *Trends in molecular medicine* (2001).

[50] THORUP, M., AND ZHANG, Y. Tabulation based 4-universal hashing with applications to second moment estimation. In *SODA* (2004).

[51] WADDELL, N., PAJIC, M., PATCH, A.-M., CHANG, D. K., KASSAHN, K. S., BAILEY, P., JOHNS, A. L., MILLER, D., NONES, K., QUEK, K., ET AL. Whole genomes redefine the mutational landscape of pancreatic cancer. *Nature* (2015).

[52] WATSON, M. Illuminating the future of dna sequencing. *Genome Biol* (2014).

[53] YAO, A. C.-C. Protocols for secure computations (extended abstract). In *FOCS* (1982).

[54] YAO, A. C.-C. How to generate and exchange secrets (extended abstract). In *FOCS* (1986).

[55] ZAHUR, S., ROSULEK, M., AND EVANS, D. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In *EUROCRYPT* (2015).

APPENDIX

A. PROOF OF THEOREM 0

PROOF. Note the D_i's ($1 \leq i \leq \ell$) computed in step 1-(a)-iii are independent identically distributed random variables and we already show $E[D_i] = E\left[(d_A - d_B)^2\right] = d$. Further, we can bound the variance of D_i. Because

$$E[D_i^2] = E[(d_A - d_B)^4] = E\left[\left(\sum_{s\in A-B} h(s) - \sum_{s\in B-A} h(s)\right)^4\right].$$

Define $h'(s)$ to be $h(s)$ for every $s \in A-B$, and $-h(s)$ for every $s \in B-A$. It is easy to verify that $E[(h'(s))^2] = E[(h'(s))^4] = 1$. Thus, define $\Delta = \mathit{Diff}(A,B)$,

$$\begin{aligned}
E[D_i^2] &= E\left[\left(\sum_{s\in\Delta} h'(s)\right)^4\right] \\
&= E\left[\sum_{s\in\Delta} h'(s)^4 + 3\sum_{s_1\neq s_2} h'(s_1)^2 h'(s_2)^2\right] \\
&= E\Bigg[\sum_{s\in\Delta} h'(s)^4 \\
&\quad + 3\left(\left(\sum_{s\in\Delta} h'(s)^2\right)^2 - \sum_{s\in\Delta} h'(s)^4\right)\Bigg] \\
&= d + 3\cdot(d^2 - d) = 3d^2 - 2d
\end{aligned}$$

Therefore $Var[D_i] = E[D_i^2] - E[D_i]^2 = 2d^2 - 2d \leq 2d^2$. Beacause D_i's ($1 \leq i \leq \ell$) are independent, we have $E\left[\hat{D}_i\right] = E\left[\frac{1}{\ell}\sum_{i=1}^{\ell} D_i\right] = d$, and that $Var\left[\hat{D}_i\right] = Var\left[\frac{1}{l}\sum_{i=1}^{l} D_i\right] \leq 2d^2/l$. Using Chebyshev's inequality, we know that

$$Pr\left\{\left|\hat{D}_i - d\right| \geq \sqrt{3}\sqrt{\frac{2d^2}{l}}\right\} \leq 1/(\sqrt{3})^2.$$

By setting $\ell = 6/\varepsilon^2$, we obtain

$$Pr\left\{\left|\hat{D}_i - d\right| \geq \varepsilon d\right\} \leq \frac{1}{3}.$$

Finally, note $X = median_{j\in[k]}\hat{D}_j$, where every $\hat{D}_j$ is bounded as above. Hence, $|X - d| \geq \varepsilon d$ happens if and only if for at least half of j, $|\hat{D}_j - d| \geq \varepsilon d$. Therefore, using a standard Chernoff bound, by setting $k = O(\lambda)$, we can get the desired bound. ∎

B. PROOF OF THEOREM 1

PROOF. In the i-th iteration, let Y_{ij} be the random variable indicator for event $g(j) = i$. We can rewrite $d_A[i] = \sum_{s\in A} Y_{is}h(s)$ and that $d_B[i] = \sum_{s\in B} Y_{is}h(s)$. Define $h'(s)$ to be $h(s)$ for every $s \in A-B$, and $-h(s)$ for every $s \in B-A$. Let $d_i = d_A[i] - d_B[i]$ and define $\Delta = \mathit{Diff}(A,B)$. Therefore

$$\begin{aligned}
d_i &= \sum_{s\in A-B} Y_{is}h(s) - \sum_{s\in B-A} Y_{is}h(s) \\
&= \sum_{s\in \mathit{Diff}(A,B)} Y_{is}h'(s) \\
&= \sum_{s\in\Delta} Y_{is}h'(s)
\end{aligned}$$

Next, we calculate the expectation and variance of d_i^2.

$$\begin{aligned}
E\left[d_i^2\right] &= E\left[\left(\sum_{s\in\Delta} Y_{is}h'(s)\right)^2\right] \\
&= E\left[\sum_{s\in\Delta}\left(Y_{is}h'(s)\right)^2\right] \\
&= \sum_{s\in\Delta} E\left[\left(Y_{is}h'(s)\right)^2\right] \\
&= \sum_{s\in\Delta} E\left[Y_{is}^2\right] E\left[h'(s)^2\right] \\
&= \sum_{s\in\Delta} \frac{1}{\ell} \times 1 = \frac{d}{\ell}
\end{aligned}$$

$$\begin{aligned}
E\left[d_i^4\right] &= E\left[\left(\sum_{s\in\Delta} Y_{is}h'(s)\right)^4\right] \\
&= E\left[\sum_{s\in\Delta} Y_{is}^4 h'(s)^4 + 3\sum_{s_1\neq s_2} h'(s_1)^2 h'(s_2)^2 Y_{is_1}^2 Y_{is_2}^2\right] \\
&= E\Bigg[\sum_{s\in\Delta} h'(s)^4 Y_{is}^4 \\
&\quad + 3\left(\left(\sum_{s\in\Delta} h'(s)^2 Y_{is}^2\right)^2 - \sum_{s\in\Delta} h'(s)^4 Y_{is}^4\right)\Bigg] \\
&= \frac{d}{\ell} + 3\cdot\left(\left(\frac{d}{\ell}\right)^2 - \frac{d}{\ell}\right) = 3\left(\frac{d}{\ell}\right)^2 - 2\frac{d}{\ell}
\end{aligned}$$

Thus we have $E[D_i] = \ell E[d_i] = d$, $Var[D_i] = \ell Var[d_i] = \ell \times (E[d_i^2] - E[d_i]^2) \leq \frac{2d^2}{\ell}$. Applying Chebyshev's and Chernoff's inequalities similar to Appendix A finishes the rest of the proof. ∎

C. PROOF OF THEOREM 2

PROOF. If a random variable $X \sim N(0,\sigma^2)$ (where $N(0,\sigma^2)$ denotes a normal distribution with expectation 0 and variance σ^2), then $Y = |X|$ follows *half normal distribution*, and further we know $E[Y] = \sigma\sqrt{2/\pi}$, and $Var[Y] = \sigma^2(1 - 2/\pi)$.

In protocol 2, assuming $d_A - d_B$ can be approximated by $N(0,d)$, the estimator for $\sqrt{d}$ is $D_i = \sqrt{\pi/2}|d_A - d_B|$, because $E[D_i] = \sqrt{d}$, $Var[D_i] = (\pi^2 - 2\pi)d/4 < d$.

With similar argument in the proof of Theorem 0 using Chebyshev's and Chernoff's inequalities, we can show that for any $\epsilon > 0, \lambda > 1$,

$$\Pr\left\{(1-\epsilon)\sqrt{d} \leq R \leq (1+\epsilon)\sqrt{d}\right\} \geq 2^{-\lambda}.$$

can be achieved with $k = O(\log(1/\delta))$, and $\ell = O(1/\epsilon^2)$. Therefore, for any $\varepsilon > 0, \lambda > 1$, by selecting an ε such that $(1+\epsilon)^2 < 1 + \varepsilon$ we can find k, ℓ to ensure $Pr\left\{|R^2 - \varepsilon| \leq \varepsilon d\right\} \geq 2^{-\lambda}$. ∎