GEMstone 2.0: Secure Interrogation of Genomic Databases (SIG-DB)

FY18 Project Report
March 3, 2018

An In-Q-Tel Labs Collaboration

Contents

Executive Summary

Genomic data are becoming increasingly valuable as we develop methods to utilize the information at scale and gain a greater understanding of how genetic information relates to biological function. Advances in synthetic biology and the low cost of sequencing are increasing the amount of privately held genomic data. As the quantity and value of private genomic data grow, so does the incentive to acquire and protect such data, which in turn creates a need to store and process these data securely.

This project explores the limitations, opportunities, and capabilities of secure computation techniques applied to DNA sequence comparisons. Using homomorphic encryption (a software-based encryption approach) the Secure Interrogation of Genomic DataBases (SIG-DB) protocol was developed to enable searches of databases (DB) of genomic sequences with an encrypted query sequence without revealing the query sequence to the database owner or any of the database sequences to the Querier. Our results show that the SIG-DB algorithm returns an accurate assessment of the similarity of queries to databases of interest. The computational runtime and information leakage were compared between a fully homomorphic approach using the Microsoft *SEAL* cryptosystem and a partially homomorphic approach using the Paillier cryptosystem. SIG-DB is the first application that we are aware of to take advantage of locality-sensitive hashing and homomorphic encryption to allow generalized sequence-to-sequence comparisons of genomic data.

We also explored an alternative approach that uses hardware-based secure computation, specifically Software Guard eXtension (SGX), by Intel®. We were unable to complete a prototype at this time due to the immaturity of the technology. However, our research findings indicate that SGX has the potential to enable a cloud-based secure computation system with, theoretically, minimal information leakage and similarity scoring execution times near equivalent to plaintext comparisons. Much research remains to be done to fully understand the operational and security limitations of the system.

We briefed government stakeholders on our prototype and findings at a recent B.Next event. Attendees expressed strong support for continued work on homomorphic encryption for secure interrogation of genomic databases. The participants provided valuable feedback on the tool and numerous use cases they encounter that could be transformed by this approach. Although the algorithm was developed specifically for microbial genomics comparisons, SIG-DB could be useful for a number of applications, including healthcare, human genomics, organizational collaborations, and more.

Background

Genomics – the ability to read, write, and edit DNA and RNA sequences– is a cornerstone of the biological revolution. Important applications of genomics during a disease outbreak include diagnostics, biometrics for patient identity, discovery of previously unknown pathogens, and mapping the origin, evolution and physical spread of pathogens. Sequencing the genome of a pathogen throughout the course of an outbreak is therefore a core capability. While the cost of sequencing has plummeted over the last decade, there still remain technical and programmatic challenges to fully exploit sequencing to combat disease outbreaks.

In FY17, B.Next initiated GEMstone, a project focused on identifying opportunities to improve analysis of pathogen genome sequences[1] and explore field-based use of sequencing during epidemics[2]. We focused on two key unsolved problems: (1) determining whether a pathogen has been genetically engineered based on signatures embedded in its genome sequence; and (2) exploring the requirements for the field use of "portable" DNA sequencing instruments to enhance epidemic management. To that end, we hosted a technology focus day and led two roundtable discussions to explore the current state of genomic analysis for outbreak management, with the goal of identifying challenge projects for FY18. During the course of our discussions with experts, we learned that some of the problems a challenge project might address include:

- Access to known, natural (unengineered) sequence data, which is increasingly privately owned and closely held.
- Access to large databases of engineered DNA sequences, which are predominantly the proprietary product of synthetic biology companies.
- Automated methods for detecting signatures of genetic engineering in genomic sequences. (B.Next focused on this challenge in FY18 project "GEMstone 2.0: Detecting Evidence of Genetic Engineering")

Accessing Biological Data

Exploiting genomic information has become an important component of the identification, characterization, and attribution of an unknown biological sample. To do so, the necessary bioinformatic analyses require known genomic data against which to compare the sample's genomic data. However, a growing amount of genomic information is privately held and proprietary. There are many reasons entities may be unwilling to share this data or make it discoverable, including IP protection and fear of liability. For example, the rapidly growing field

[1] The report from project GEMstone can be found on the B.Next website: https://www.bnext.org/article/roundtable-interrogation-of-suspect-biological-samples/

[2] The report from the portable sequencing roundtable can be found on the B.Next website: https://www.bnext.org/article/roundtable-portable-sequencing/

of synthetic biology is transforming the marketplace ($3.9B in 2016; projected to grow to $11.4B by 2021[3]). In particular, synthetic biology is transforming industrial practices, where microorganisms are exploited to produce commercial products, resulting in customized microbes and unique genomic data. With that comes fierce corporate competition, and the genomic information becomes proprietary.

The proliferation of private DNA sequence databases could hinder the US Government from comparing their sequence data to a large portion of known DNA sequences, both native and engineered. The current process for government agencies to query a privately held database requires a complex legal agreement between the agency and the private entity. Furthermore, due to the security concerns around releasing information about a sample under investigation, the private entity will likely need a security clearance or have cleared personnel to work with the government investigation. This is because no encryption techniques or secure hashing of query sequences are currently in use to enable masking of the query sequence.

Need for Secure Genomic Computation Methods

Data in storage can be protected by hardware- or software-based encryption, but sensitive data is vulnerable during processing. Therefore, there is a need for methods, including search and comparison methods, which permit the processing of data while preserving security. Secure genomic computation methods would allow the biodefense community (and government, broadly) to explore, and potentially use, the genomic information being generated and held in the private sector. This capability could reduce the time to respond to a biological event, whether natural or deliberate, by increasing the quantity of data used during critical bioinformatics analyses, and our understanding of it. Additionally, secure genomic comparison methods might enable government agencies to exploit data at other agencies or within their own agency that is currently unreachable because of differing security classifications.

The utility and impact of secure genomic computations goes beyond biodefense. Genomic information is becoming valuable to a variety of applications, including healthcare. There is inherent tension between the Health Insurance Portability and Accountability Act (HIPAA) requirements for patient data privacy, and the public health or scientific utility of genomic data. For example, researchers who want to study the genetics of disease may wish to compare patient genome databases in a way that allows data mining while complying with HIPAA obligations.

[3] Flores Bueso Y, Tangney M. Synthetic Biology in the Driving Seat of the Bioeconomy. Trends Biotechnol [Internet]. Elsevier; 2017 Nov 16;35(5):373–378. Available from: http://dx.doi.org/10.1016/j.tibtech.2017.02.002

Because human genomes are highly conserved, much of the existing work is on methods for assessing genomic variation at the single nucleotide polymorphism[4,5] or causal variant[6] level.

The impact of the microbiome on human health[7,8] is an active area of research. As we expand our understanding of the microbiome, our personal microbial fingerprint may become part of the data stored in electronic health records[9]. Therefore, we may arrive at a consensus that a person's microbiome is as important to protect as a person's genome.

Project Overview

Project SIG-DB is a collaborative project that engaged expertise from all four IQT Labs (B.Next, Lab41, CyberReboot, and CosmiQ Works). We explored the application of secure computational information sharing techniques to enable a "query in place" approach while still maintaining the privacy of both query and database entities. We focused on two approaches: **(1) Homomorphic Encryption** – software based, computational techniques to mask both the query sequence and returned results; and **(2) Security Enhanced Third Party** – a private entity that enables direct comparison of sequences using a cloud-based environment equipped with Software Guard eXtension (SGX). SGX is a commercial, hardware-based security solution that protects code and data using protected areas of execution in memory (enclaves).

We developed a proof-of-concept algorithm that leverages homomorphic encryption to demonstrate the feasibility of such an approach. The opportunity, advantages, and limitations of the Security Enhanced Third Party approach were also explored and our informed opinion of its applicability is included in this report.

---

[4] Kim M, Lauter K. Private genome analysis through homomorphic encryption. BMC Med Inform Decis Mak. BioMed Central; 2015;15(5):S3.

[5] Ziegeldorf JH, Pennekamp J, Hellmanns D, Schwinger F, Kunze I, Henze M, Hiller J, Matzutt R, Wehrle K. BLOOM: BLoom filter based oblivious outsourced matchings. BMC Med Genomics. 2017 Jul;10(2):44. PMID: 28786361

[6] Jagadeesh KA, Wu DJ, Birgmeier JA, Boneh D, Bejerano G. Deriving genomic diagnoses without revealing patient genomes. Science (80-). 2017 Aug 18;357(6352):692 LP-695. PMID: 28818945

[7] Goodrich JK, Davenport ER, Clark AG, Ley RE. The Relationship Between the Human Genome and Microbiome Comes into View. Annu Rev Genet [Internet]. Annual Reviews; 2014 Jan 13; Available from: https://doi.org/10.1146/annurev-genet-110711-155532

[8] Hall AB, Tolonen AC, Xavier RJ. Human genetic variation and the gut microbiome in disease. Nat Rev Genet [Internet]. Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved.;; 2017 Aug 21;18:690. Available from: http://dx.doi.org/10.1038/nrg.2017.63

[9] Chuong KH, Mack DR, Stintzi A, O'Doherty KC. Human Microbiome and Learning Healthcare Systems: Integrating Research and Precision Medicine for Inflammatory Bowel Disease. OMICS. United States; 2017 Mar; PMID: 28282257

Methods

Assumptions

For this project, the team made the following assumptions prior to developing the algorithm:

- The SIG-DB protocol will be used between two parties: a Querier and a Database Owner.
- The Database Owner is "willing-but-unable" to share their genomic data without having a way to protect their data.
- The Querier is not intending to hide the fact they are searching a Database. This protocol is not intended to be used without the Database Owner being involved in the execution.
- Neither the Querier nor the Database Owner want to divulge any sequence data. However, it may be acceptable if some information is learned about the database, such as number of entries in the database, maximum length of sequences in the database, and the level of diversity among the sequences in the databases.
- It is also expected that the time to result does not need to be within minutes. An acceptable time to result (from start to finish) could be hours to days, depending on the specific use case.
- For highest impact, no (or limited) expertise in encryption or genomics is required to operate the tool. However, the user should have basic computing skills equivalent to operating standard commercial software.
- The Querier will have their own policies or procedures in place for what to do in response to the tool's reported similarity scores. For example, if a sufficiently similar sequence is found based on the threshold set by the Querier, then a possible next step in a forensic investigation would be to get a warrant to obtain the database.
- Both the Querier and the Database Owner have compute infrastructure to execute the encryption/decryption and the scoring, respectively.

Selecting Approaches for Secure Computation

Software-based approaches. Several approaches have been published that demonstrate software-based, secure, multi-party computation, including Yao's protocol,[10,11] secure fault-tolerant protocols, [12] oblivious transfer,[13] and many others. In recent years, the field of secure computation has shifted towards improving the performance of such methods for specific applications (e.g. electronic voting, electronic auctions, sharing of signature or decryption functions, private information retrieval, etc…), largely driven by the computational expense that

[10] Yao AC. Protocols for Secure Computations. IEEE 1982. http://research.cs.wisc.edu/areas/sec/yao1982-ocr.pdf
[11] Yao AC. How to generate and exchange secrets. Foundations of Computer Science 1986. DOI:10.1109/SFCS.1986.25
[12] Galil A, Haber S, Yung M. Cryptographic Computation: Secure Falut-Tolerant Protocols and the Public-Key Model. Advances in Cryptology 1988: p135-155. https://link.springer.com/content/pdf/10.1007%2F3-540-48184-2_10.pdf
[13] Kilian J. Founding cryptography on oblivious transfer. Proceedings of the twentieth ACM symposium on Theory of computing 1988: p20-31. DOI: 10.1145/62212.62215

has limited the use for many of these protocols. Several approaches could be chosen for the application we are focused on in this project (and others beyond what we tried should be explored), but we chose to focus on homomorphic encryption for the software-based approach.

Homomorphic encryption (HE)[14,15,16, 17] is a unique class of methods that allows a piece of information to be encrypted and shared, operated upon, and an answer returned without ever revealing the underlying information. The encrypted data can only be revealed to the owner using their private key. It represents a method of growing interest to the private sector because of its utility in enabling multi-party interactions, and as such we believe a thorough understanding of the opportunities it presents for biological applications is essential for IQT and the USG.

Homomorphic encryption can be separated into two classes: fully HE and partial HE. Fully HE systems allow an arbitrary set of operations (e.g., addition, subtraction, division, and multiplication) to be performed on ciphertext. Partial HE systems allow a much smaller subset of operations on ciphertext, with each encryption system developed to support a specific set of operations. The Paillier homomorphic encryption system (PHE) [18] is a partial, additive-only (+) HE cryptosystem. PHE algorithms can be orders of magnitude faster than fully HE cryptosystems[19] and thus are promising for near-term operational applications. To date, there is no published work combining any version of HE applications with polynucleotide sequence-to-sequence comparisons, that we are aware of.

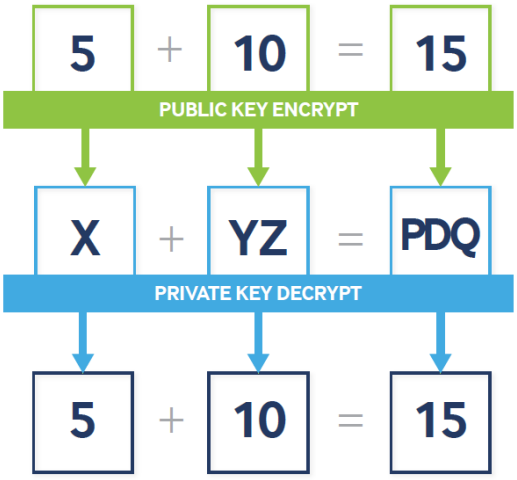

Figure 1: Homomorphic encryption – operations on plaintext are mirrored in cipher text space.

Hardware-based approaches. There are a number of hardware-based options for data security at rest, including AES full disk encryption (FDE) now available as hardware options by popular storage device manufacturers such as Seagate®, Hitachi, Western Digital, Samsung, PureStorage (an IQT Portfolio company) and EMC, just to name a few. However, these options do not secure

[14] Kim M, Lauter K. Private genome analysis through homomorphic encryption. BMC Med Inform Decis Mak. BioMed Central; 2015;15(5):S3.
[15] Ziegeldorf JH, Pennekamp J, Hellmanns D, Schwinger F, Kunze I, Henze M, Hiller J, Matzutt R, Wehrle K. BLOOM: BLoom filter based oblivious outsourced matchings. BMC Med Genomics. 2017 Jul;10(2):44. PMID: 28786361
[16] Dowlin N, Gilad-Bachrach R, Laine K, Lauter K, Naehrig M, Wernsing J. Manual for using homomorphic encryption for bioinformatics. Proc IEEE. IEEE; 2017;105(3):552–567.
[17] Gentry C. Fully homomorphic encryption using ideal lattices. STOC. 2009. p. 169–178.
[18] Paillier P. Public-key cryptosystems based on composite degree residuosity classes. Eurocrypt. Springer; 1999. p. 223–238.
[19] Chen H, Laine K. Simple Encrypted Arithmetic Library - SEAL v2.2 [Internet]. 2017 Jun. Available from: https://www.microsoft.com/en-us/research/publication/simple-encrypted-arithmetic-library-seal-v2-2/

sensitive data while it resides in memory being processed. A relatively 'new-to-the-scene' technology that could enable secure data operations is a recent addition to the Intel® instruction set architecture (ISA) called Software Guard eXtension (SGX). SGX enables a secure enclave to be created and accessed only by trusted processes, permitting operations on sensitive data in situations where the system would otherwise be untrusted. The SGX-capable executable can be run alone since the CPU itself provides the encrypted environment. Although performance degradation is not well-known at this time, computational overhead is expected to some degree given the complex protocols to go 'in' and 'out' of the enclave and the need to encrypt/decrypt. It is worth noting that SGX and HE do not need to be viewed in an "either/or" fashion; rather, one might choose to implement both. The addition of a partial or fully homomorphic encryption to SGX would add additional layers of data protection by obscuring the data itself, thus reducing the impact of exposure.

SIG-DB protocol description

SIG-DB can be broken down into four major steps: (1) sequence pre-processing, (2) query encryption, (3) scoring, and (4) decryption. Steps 2 and 4 utilize whichever encryption approach is desired (PHE or FHE in our case). Steps 1 and 3 remain the same regardless of the selected encryption approach.

Sequences (query and database entries) are pre-processed prior to encryption or scoring by splitting into k-mers, applying a hash function, and then converting into a locality sensitive hash (LSH). The use of k-mers is an established method of breaking up genetic sequences without appreciative data loss. The size of the k-mer is a tunable parameter that can be optimized for a specific application. LSHs are also space efficient data structures that store information at either a 1 or 0 and, unlike bloom filters, force the data to a preset vector size that is smaller than the original data dimensions. For example, the sequence 'ATCGGATC' would represent a 1 in a separate location within the LSH than 'GTAACAGT' (see Figure 2). The size of the LSH is set based on the size of the data being stored and the pre-determined limit for hash collisions. For our application, the LSH length is set to be five times the length of the longest sequence of the sequences being scored, thus the probability of a hash collision is 18%. This probability can be changed based on user requirements, and lowered by increasing the size of the LSH (with an increase in runtime), or by using multiple hash functions in conjunction to build a bloom filter rather than an LSH[20]. Using this pre-processing approach, we can reduce data size, and by extension, compute time.

[20] Bloom BH. Space/Time Trade-offs in Hash Coding with Allowable Errors. Communications of the ACM. 1970: 13(7):422-426. http://dmod.eu/deca/ft_gateway.cfm.pdf

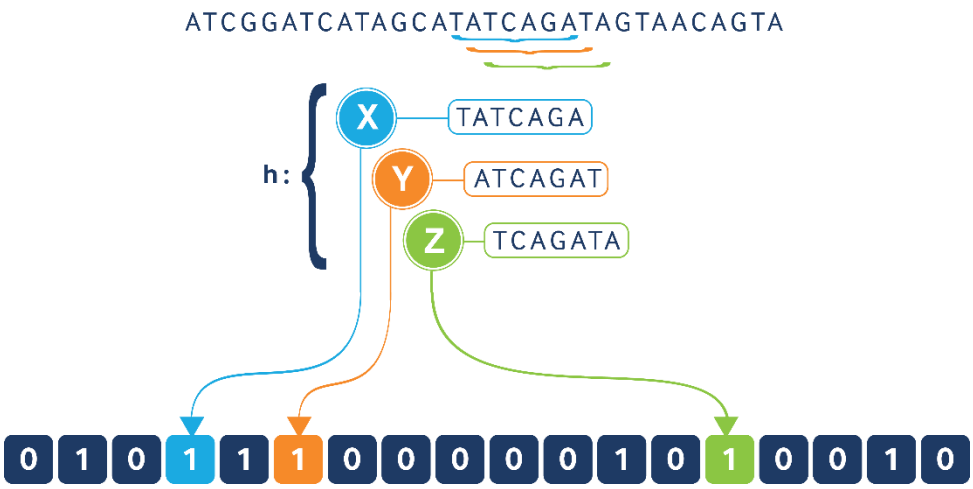

Figure 2: Example of a Locality Sensitive Hash (LSH), derived from a DNA sequence. In SIG-DB, the k-mers are created using a sliding window of 1 character, as illustrated, with a sequence of length n resulting in 'n-k' k-mers.

Similarity scores are calculated as Intersection over Union (IoU), Intersection over length of Query (IoQ), and Intersection over length of Database entry (IoD) (see Figure 3). The IoU is a common metric when looking at the proportion of an item correctly classified, such as in computer vision. This metric is efficient to compute and well established; therefore, it is the prime candidate for our application. IoD and IoQ were included to provide an alternative metric that would account for sequence length variation between the query and database sequences. Additional metrics could be useful, but a full characterization of metrics is beyond the scope of this project.

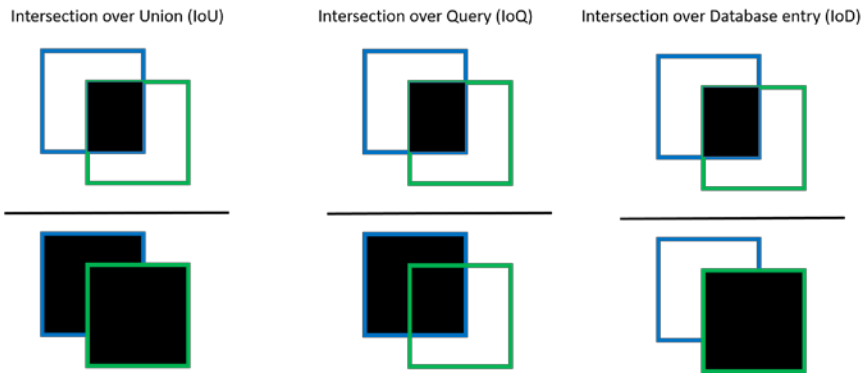

Figure 3: Visual depiction of the similarity scores: IoU, IoQ, and IoD. The blue boxes represent the query LSH, and the green boxes represent the database entry LSH. The element included in the calculation is notated by the black box.

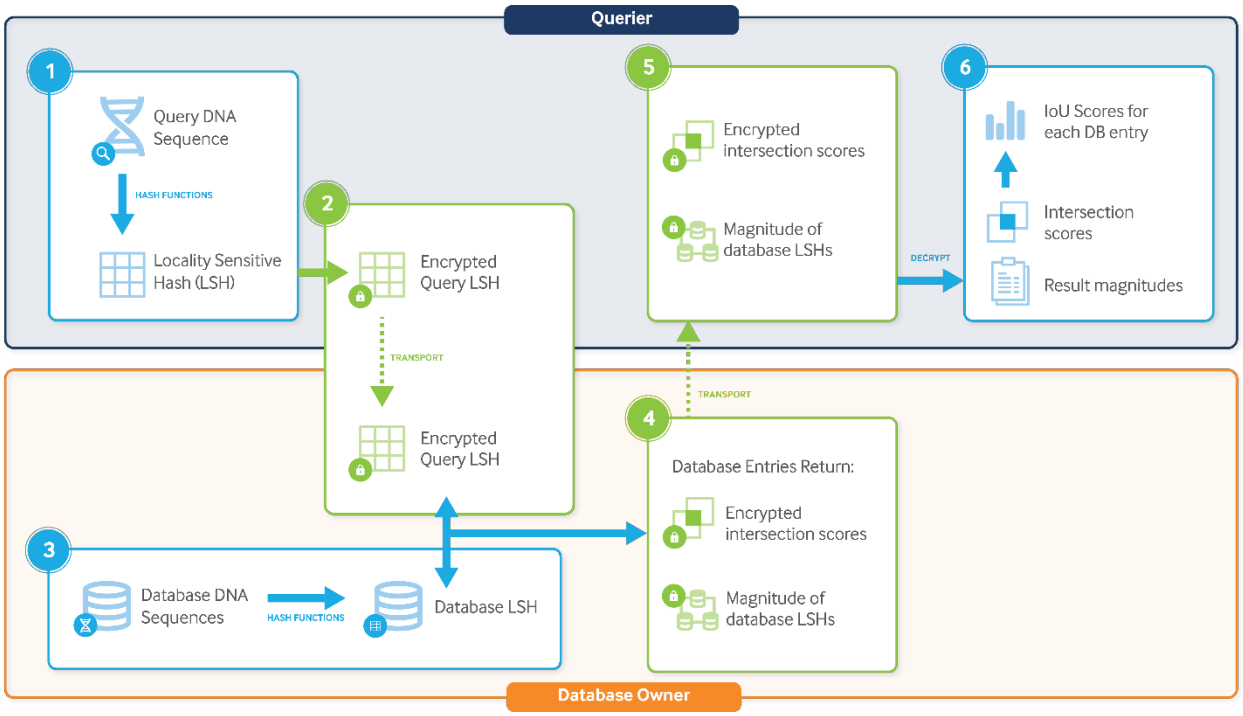

Figure 4: SIG-DB protocol for (1) hashing sequence into locality sensitive hash (LSH), (2) encrypting and passing LSH, (3) hashing database elements into LSHs, (4) comparing encrypted query LSH to database LSHs, (5) passing scores and decrypting, and (6) calculating IoU, IoD, and IoQ scores.

The SIG-DB protocol (whether PHE, FHE, or SGX) is executed as follows (steps below coincide with steps in Figure 4):

(1) The Querier converts the query sequence to a Locality Sensitive Hash (LSH)
(2) The Query LSH is encrypted and transported to the Database Owner, along with the scoring executable, hash function (same one used by Querier), and instructions.
(3) The Database Owner uses the hash function to create an LSH for each entry of the database. NOTE: the Database LSHs do not need to be encrypted, as Database records never leave the Database firewall.
(4) The Database Owner uses the executable to compare the encrypted Query LSH to the Database LSHs. An encrypted intersection score is generated for each Database entry. The magnitude (or sum of all '1's in the result sequence LSH) is also computed for each Database entry; this value is unencrypted in the current implementation[21].
(5) The encrypted intersection scores and unencrypted computed magnitudes are returned to the Querier.
(6) The Querier decrypts the intersection scores and calculates the similarity scores (IoU, IoQ, and IoD) using the computed magnitudes and the following equations:

[21] It is a negligible algorithm extension to implement encryption for transport of the Database entry magnitude. However, this would substantially increase runtime.

$$IoU = \frac{Intersection}{|Query\ LSH| + |DB\ entry\ LSH| - Intersection}$$

$$IoQ = \frac{Intersection}{|Query\ LSH|} \qquad IoD = \frac{Intersection}{|DB\ entry\ LSH|}$$

Figure 5: Definitions of similarity scores: IoU, IoQ, and IoD

Results

We constructed a series of tests to evaluate the build and performance of the algorithm:

(1) Parameter assessment – how does the size of the sequence and k-mer, as well as sequence mutation and location/density of mutations, affect the results?
(2) Correctness – how well does the algorithm return the correct results, based on ground truth data?
(3) Practicality - what are the trade-offs between query length, database size, number of cores used, and time of execution?

Data

We accessed and downloaded bacterial genome sequences from the National Center for Biotechnology Information (NCBI) Assembly RefSeq[22] database, a public repository of validated, reference-quality sequences deposited by researchers and scientists and confirmed by NCBI scientists. Our dataset contained a collection of complete genomes, chromosomes, scaffolds, and contig[23] sequences for all available bacterial species, totaling 75,958 sequences and 244 gigabytes of data. A subset of sequences from this database were selected for the tests described below. Queries were selected out of each respective test dataset.

Parameter Assessment

We conducted tuning experiments to identify the effect of k-mer size on algorithm performance, and tested the extent to which SIG-DB is robust to sequence mutations, using a dataset of 50 *E. coli* genomes and 50 *S. aureus* genomes. The k-mer length was tested at k={8, 16, and 32}[24]. Uniformly distributed, random mutations were introduced to query sequences *in silico* in 5% increments, ranging from 0%-100%. Using a k-mer length of 8, the SIG-DB algorithm correctly returned the sequence of interest as the highest IoU score for sequences with mutation rates from 0-45% for *E. coli* and 0-35% for *S. aureus.* The mutation rate tolerance decreases as the k-mer size increases, as shown in Figure 6. We found that the method correctly identified the

[22] Coordinators NR. Database resources of the National Center for Biotechnology Information. Nucleic Acids Res [Internet]. Oxford University Press; 2016 Jan 4; 44 (Database issue):D7–D19. Available from: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4702911/

[23] A contig is a set of overlapping DNA sequences (e.g. sequence reads) that together form a consensus sequence for a region of DNA.

[24] These values were selected based on the optimization results of various bioinformatics approaches found in the literature.

presence of similar sequences in the subject database to a greater level of divergence when k was set to 8. Therefore, we set k={8} for all subsequent work.

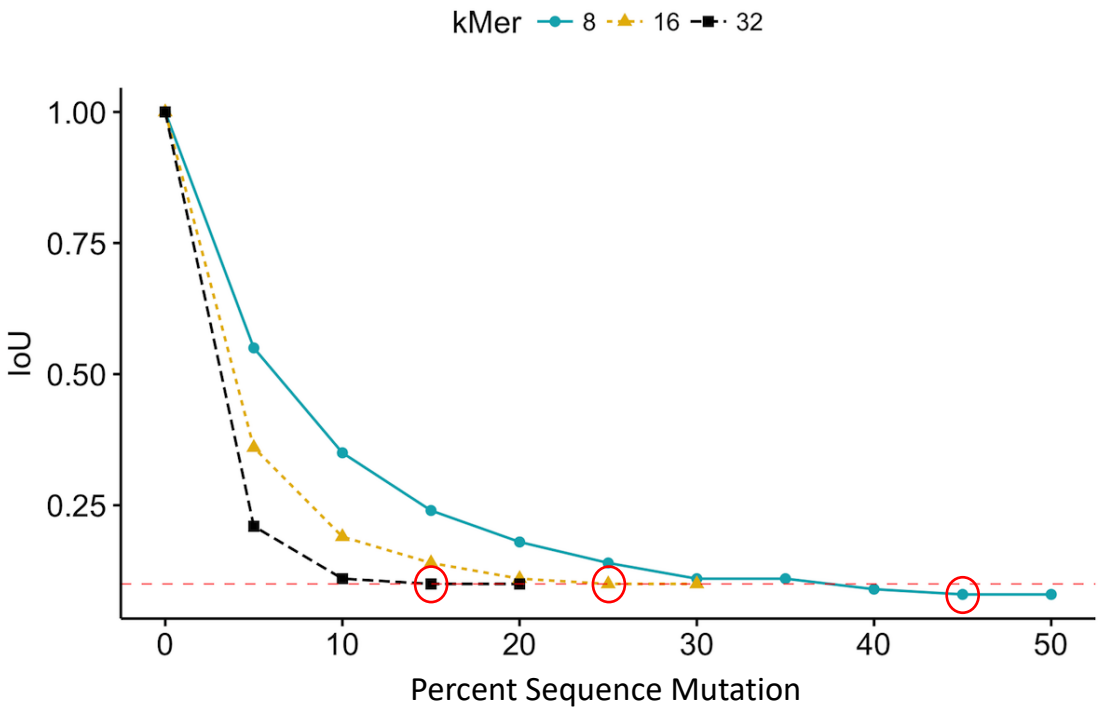

Figure 6: Intersection over Union (IoU) scores for k-mer = {8, 16, 32} with random query sequence mutation rates from 0-100%. K-mer = 8 showed best performance, with correct identification of sequence of interest up to mutation rates of 45%. The red circles indicate the largest mutation rate for each k-mer size that returned the correct result as the highest IoU value. The horizontal red line represents the lowest IoU value where all 3 k-mer sizes returned the correct result.

In biological systems, mutations are not always uniformly distributed across a sequence[25], but instead may be clustered to one segment of a sequence[26]. To test how the scores are affected by this later scenario, the query sequences were randomly mutated using the same methodology as above, but localized to the first half of the sequences. The mutations ranged from 0%-100% of the first half of the sequence, representing total sequence mutations of 0%-50%. As shown in Figure 7, the similarity scores had a less dramatic decline when mutations are localized to a section of the sequence, compared to the sharp decline that occurred when the mutations are dispersed throughout the entire sequence. This observation is expected, because fewer k-mers are affected by the mutations when they are localized to one half of the sequence.

[25] Random distribution of point mutations across the entire sequence could be caused by natural genetic drift over time or intentional mutagenesis in genome editing experimentation.

[26] Portions of a genome, such as a gene, could be different between two genomes due to horizontal gene transfers, recombinant events, or distant evolutionary relations. These changes are also possible with intentional genome editing experimentation.

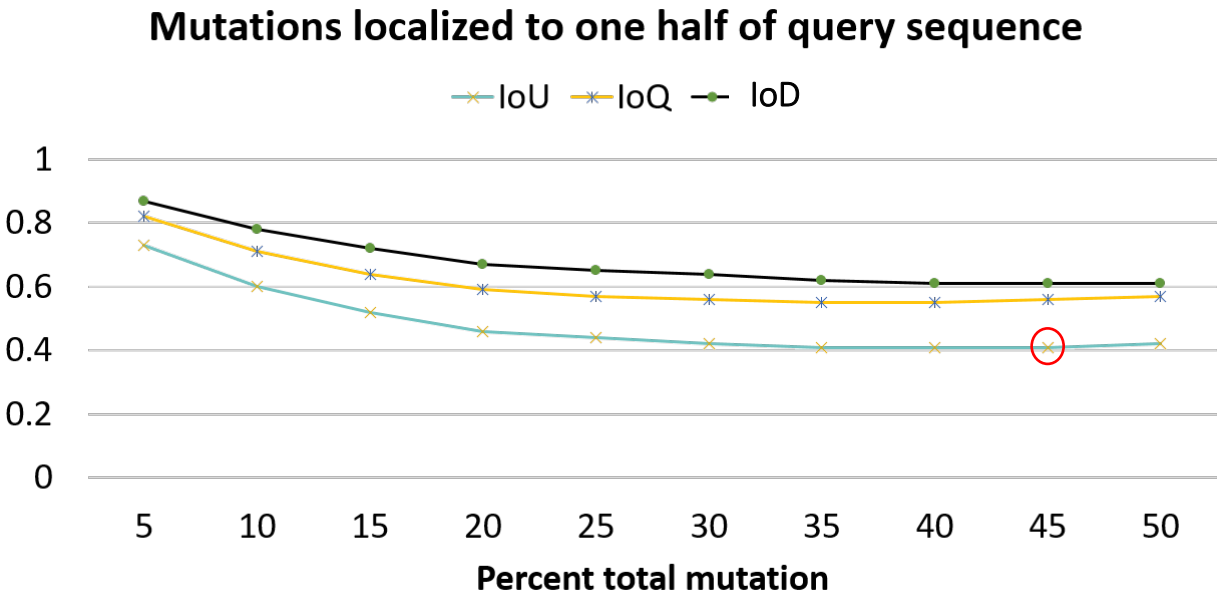

Figure 7: Similarity scores (IoU, IoQ, and IoD) for k-mer = {8} with random sequence mutation localized to one half of the sequence at rates from 0-100%. The red circle represented the largest mutation rate that correctly returned the sequence of interest as the highest IoU score.

Biological sequences inherently are quite variable in size, with genomes ranging from thousands (viruses) to billions (plants and mammals) of basepairs, depending on the species. Therefore, it is likely the query sequence will not be the same size as the database entries. To test the robustness of SIG-DB and the similarity metrics for queries and database entries of varying relative lengths, we compared performance on sequences that ranged in relative length from 1:1 to 4:1/1:4 (Q:DB, respectively). The sequence lengths tested varied from 5,000bp to 20,000bp, which represent the typical size ranges for viral genomes and bacterial genes. The k-mer size was held constant at k={8}, and no mutations were introduced. For all ratios tested, SIG-DB returned the correct sequence as the highest IoU; results shown in Figure 8. When the database sequences were held constant at 20,000bp and the query sequence reduced to smaller sizes, the IoQ=1.0 for every test. When flipped, the IoD=1.0 for every test. This performed as expected, because for IoQ (or IoD), only the length of the query (or database entry, respectively) is factored into the similarity score calculation. Therefore, the scoring would be a perfect match for all ratios tested. This demonstrates the importance of including all 3 metrics in the output, to account for sequence size variations.

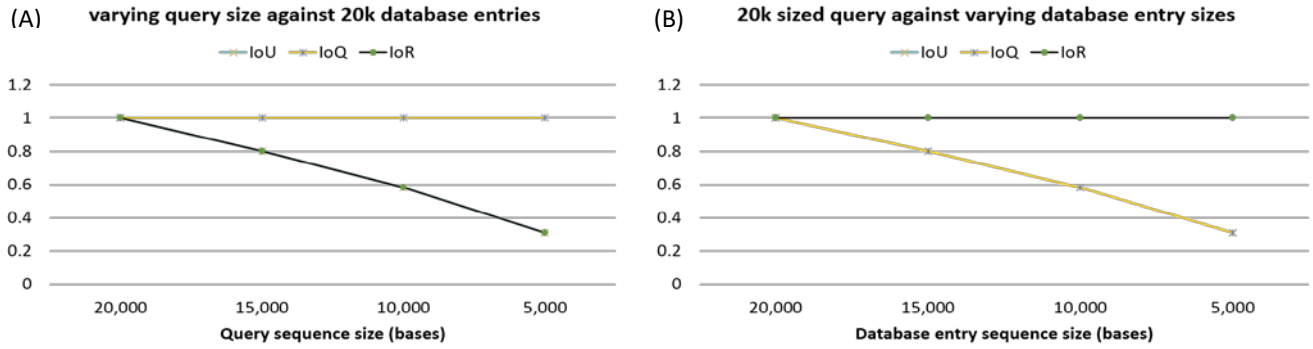

Figure 8: Similarity scores for varying sizes of query and DB entries. (A) DB held constant at 20,000 bases while query varied from 5,000-20,000 bases. (B) Query held constant at 20,000 bases while DB varied from 5,000-20,000 bases.

Correctness

The results of these parameter tests indicate that the algorithm is functioning properly, is robust against mutations (exact tolerance depends on parameters selected), and can tolerate variation in query and database sequence lengths. Next, we wanted to test the robustness of the algorithm to return the correct results. In order to measure this, we conceptualized the algorithm as a classifier and tested the classification accuracy by setting a threshold, above which the algorithm can be thought of as reporting that a particular database entry was similar to the query. This allowed us to determine the tradeoff between True Positive Rate (TPR) and the False Positive Rate (FPR) - also known as the Receiver Operator Characteristic (ROC).

We elected to use Average Nucleotide Identity (ANI) as ground truth. ANI is a measure of pairwise bidirectional nucleotide similarity, and is commonly used in bioinformatics as an *in silico* proxy for *in vitro* DNA hybridization experiments (the gold standard for sequence similarity comparisons). In our experiments, ANI was computed for each query-database entry pair using the ANIb algorithm[27].

The ANI algorithm returns a percent identity score for two sequences. In order to use ANI as a source of 'true negatives' and 'true positives' we needed to select a threshold value. It is worth noting that this threshold selection itself introduces another layer of sensitivity-specificity tradeoff, but for our purposes we chose to investigate only two specific points: 99% and 95%. These thresholds were chosen based on recent studies that demonstrated 95% and 99% are

[27] Richter M, Rossello-Mora R (2009) Shifting the genomic gold standard for the prokaryotic species definition. Proc Natl Acad Sci USA 106: 19126-19131. doi:10.1073/pnas.0906412106

scientifically accepted limits to determine species or isolate level differentiation, respectively[28,29].

A dataset was created from two *E.coli* genomes (K-12 MG1655 and O157:H7, taken from the RefSeq database). 500 non-overlapping database entries were created by sampling at random from the full genomes for each of three lengths: 1000, 2000, and 3000 bp. From each database entry, a 1000 bp query sequence was selected at random. (Queries taken from database entry of a given length were used in searches against the database set from which they were generated [30,31].) In order to have sufficient coverage of the desired genetic diversity, the selected query sequences were duplicated twice and the duplicates randomly mutated, where each base pair in the two duplicates had a 5% or 10% chance (respectively) of being switched to an alternate base. In total, this produced 1,500 queries for each length-set of database entries: 500 wild-type, 500 samples with 5% mutation, and 500 samples with 10% mutation.

ANI percent identity and SIG-DB similarity scores were computed for each database entry-query pair within a given length-set. For the score comparison, the maximum of the IoU, IoQ, or IoD (referred to here as Similarity score) was taken to account for sequence:query length differences. (The ROC curves for IoU, IoQ, and IoD can be found in Appendix 1, which further illustrate the importance of calculating all three scores. The ANI data statistics can also be found in Appendix 1.) For the ANI tests using 95% and 99% ANI, a binary classification was created such that "Relevant" sequences had an ANI ≥ 95% and ≥ 99% respectively, and "Not Relevant" sequences had an ANI < 95% and < 99% respectively. To generate the ROC curve, multiple classification thresholds were defined for the SIG-DB Similarity scores such that all sequences with Similarity ≥ threshold were considered "Relevant". The thresholds were set in an increasing manner, starting with Similarity = 0 representing a threshold that counted all sequences as "Relevant". Each subsequence threshold was then set as the next highest Similarity score in the result set. For example, if five sequences had Similarity scores of [0.2, 0.33, 0.33, 0.75, 0.87] then each of the unique scores would be set as a classification threshold, resulting in the following counts of the number of "Relevant" sequences at each threshold: [5, 4, 2, 1]. The following definitions were used to measure performance:

---

[28] Goris J, Konstantinidis KT, Klappenbach JA, Coenye T, Vandamme P, Tiedje JM (2007) DNA-DNA hybridization values and their relationship to whole-genome sequence similarities. Int J Syst Evol Microbiol. 57: 81-91, doi: 10.1099/ijs.0.64483-0

[29] Kim M, Oh H, Park S, Chun J (2014) Towards a taxonomic coherence between average nucleotide identity and 16S rRNA gene sequence similarity for species demarcation of prokaryotes. Int J Syst Evol Microbiol. 64: 346-351, doi: 10.1099/ijs.0.059774-0

[30]Queries were created from the database entries to ensure we would get both positive and negative hits. Given that each position can be one of 4 options [A, T, C, G] and we are using long sequences [500-20,000+], the likelihood of a sequence reaching 70% by chance is so small that we do not worry about accidental matches at this similarity level.

[31]It is widely accepted in the scientific community that sequences with approximately 70% or less similarity are likely not related. Therefore, discerning below that level is not likely of interest for biological applications. .

- True Positive (TP) = the number of scores classified as “Relevant” by both the Similarity and ANI scores.
- False Positive (FP) = the number of scores classified as “Relevant” by the Similarity score and “Not Relevant” by the ANI score.

For each new threshold, the True Positive and False Positive rates were calculated, then plotted as the ROC curve (see Figure 9).

The algorithm performed well across all database entry length sets and under both ANI thresholds. In fact, with an ANI threshold of 0.95 (a slightly more ‘permissive’ gold standard), the algorithm was essentially a perfect classifier - assigning Similarity scores to database entry-query pairs in exactly the same order as their ANI score rankings.

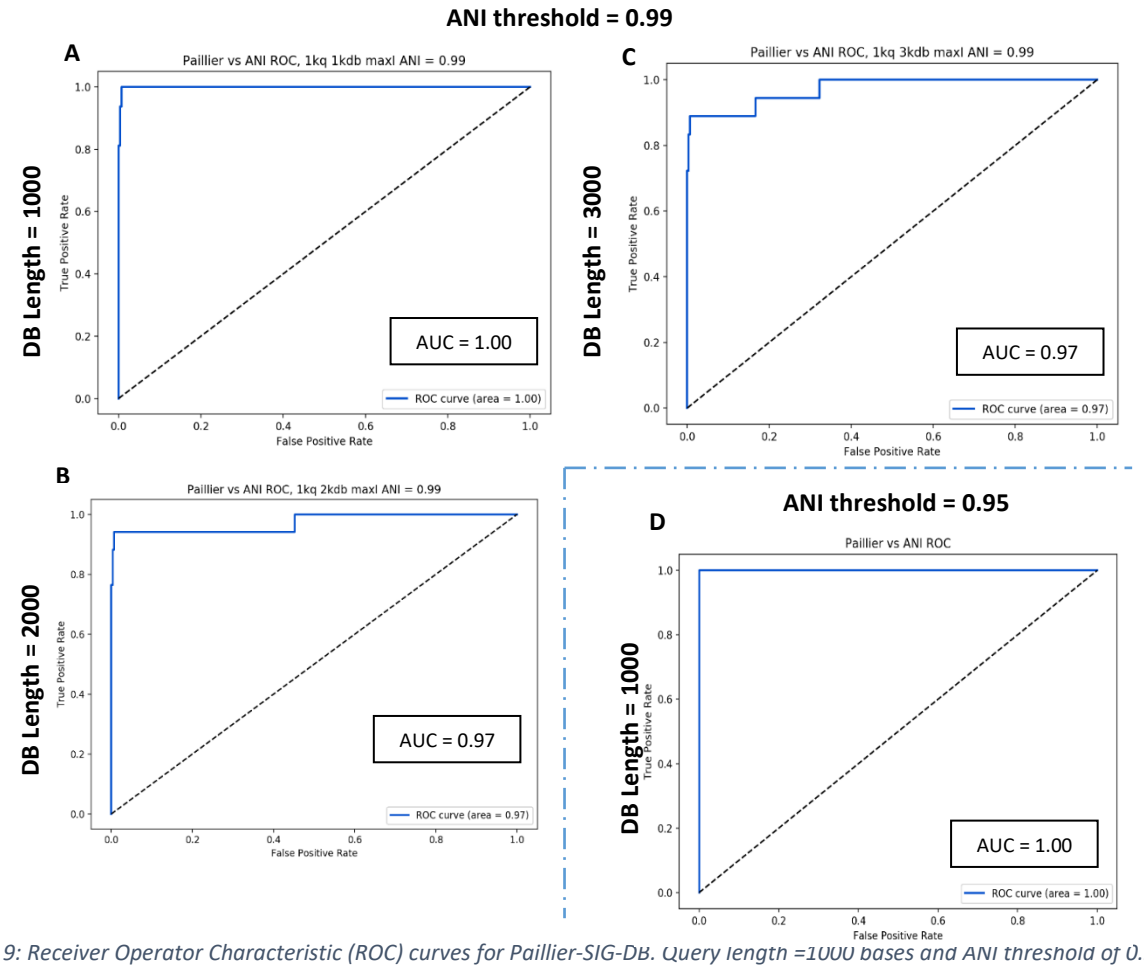

Figure 9: Receiver Operator Characteristic (ROC) curves for Paillier-SIG-DB. Query length =1000 bases and ANI threshold of 0.99 for A-C; 0.95 for D. (A) DB entry length=1000 bp with a total of 750,000 query to database entry comparisons and 1,515 of those with ANI scores above the ANI threshold (true positives) (B) DB entry length=2000 bp with a total of 750,000 query to database entry comparisons and 1,549 of those with ANI scores above the ANI threshold (C) DB entry length=3000 bp with a total of 750,000 query to database entry comparisons and 1,708 of those with ANI scores above the ANI threshold (D) DB entry length=1000bp with a total of 750,000 query to database entry comparisons and 1,877 of those with ANI scores above the ANI threshold (representative of all ROC curves generated using an ANI threshold=0.95).

Practicality

One major concern for using homomorphic encryption techniques for operational purposes is the computational expense. To assess the practicality of using SIG-DB for biodefense applications, we performed a series of tests to understand:

- The amount of time required to run the full operation on a single CPU core
- The time savings that could be achieved using a parallelized approach
- The breakout of time requirements by step in the protocol (encryption time, query time, and scoring time)

All bacterial sequences in the RefSeq database were used to perform these tests. The database records have a minimum length of 337 bases, a maximum length of 12,106,419 bases, and an average length of 2,756 bases. Heat maps were generated from test runs that evaluate 3 different variables: (1) number of cores used for computation (1 - 48 cores), (2) query length (100 – 20,000 bases), and (3) number of entries in the database (100, 1000, 5000, and 10000 FASTA[32] files; or 10,145 sequences, 89,992 sequences, 425,772 sequences, and 860,984 sequences, respectively).

Of Note:

- For the *practicality* tests only, the SIG-DB algorithm included an additional step that chopped each database entry into equal-sized pieces as the query sequence. Therefore, for all *practicality* tests, query length = database entry length.
- SIG-DB was written in Python®, which is not optimized for computational performance. It is expected the execution time would improve if SIG-DB was re-developed and optimized in C++ or equivalent code.

[32] FASTA is a text-based file format commonly used in bioinformatics applications.

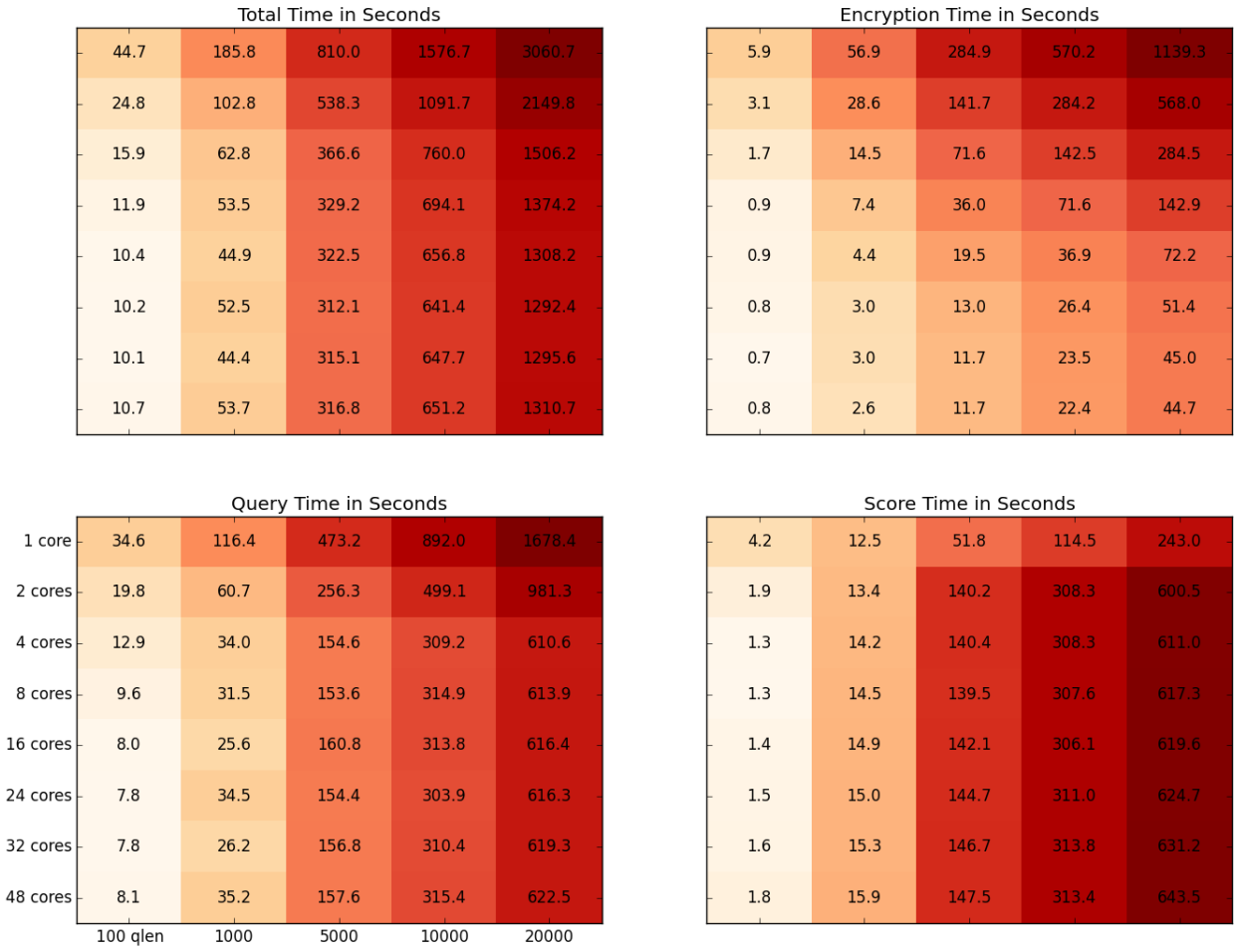

Figure 10: Heat Map showing time of execution for SIG-DB, comparing a single query of 'qlen' length to a database containing 1000 entries (82,992 sequences). The trends in this heat map are representative of all scenarios tested.

Observation #1: Parallelization helps performance, but has diminishing returns for more than 16 CPU cores. The reduction of runtime is evident by the heat maps in Figure 10 (*Scoring Time* is the exception; see Observation #3 below). For example, with a 20,000 bp sequence length and a database of 82,992 sequences, the execution time goes from 51 minutes with 1 core to 21 minutes with 16 cores - a reduction of more than 50%. However, the execution time is nearly the same for the test runs using 16 or more cores (Figure 11). The dwindling improvement beyond 16 cores likely indicates that 16 cores are sufficient infrastructure to carry out the query efficiently, and that the cost of splitting the query into smaller operations (across more cores) is greater than the performance gained – often referred to as over-parallelization. This also implies that as the dataset increases in size, the value of more CPUs could increase beyond 16 cores; however, that increasing value is not linear.

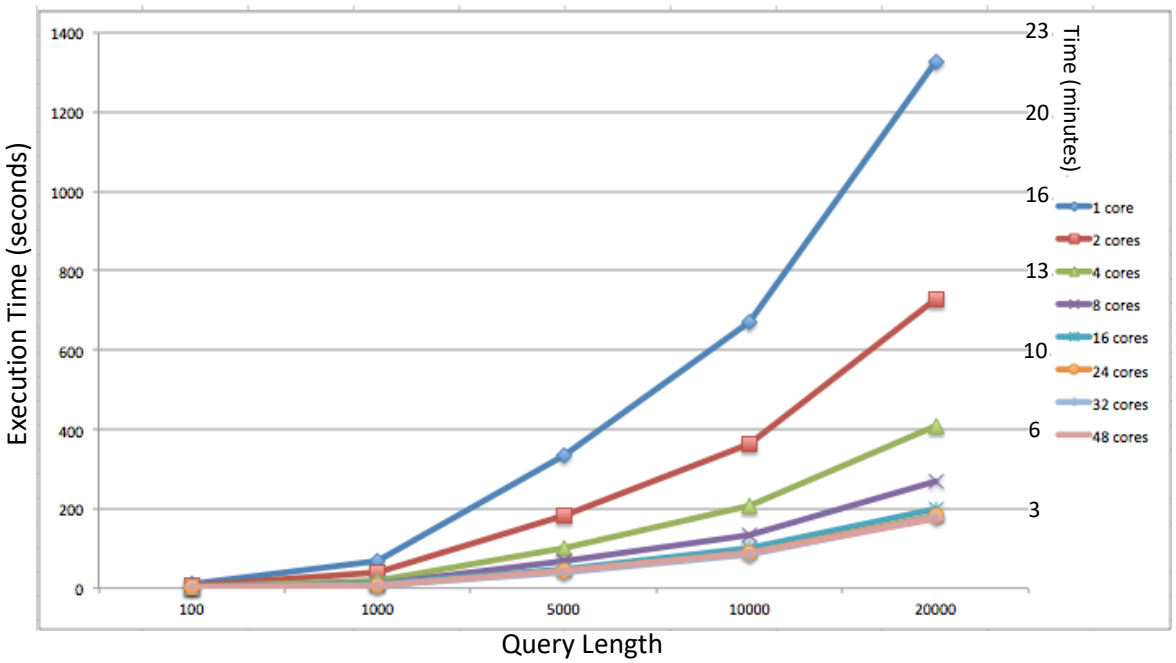

Figure 11: SIG-DB Execution time for a query against a database of 100 entries (10,145 sequences) using various numbers of cores for computation.

Observation #2: *Query Time* and *Scoring Time* are the most time intensive steps, and they appear to be roughly equal, even as parameters change. This is to be expected, because an encrypt/decrypt operation is conducted on each element of the database.

Observation #3: *Scoring Time* performs worse when parallelized. While the root cause of this is still unknown, it is likely related to the size of the query being scored and how efficient it is, or is not, to divide the work over several CPUs at the same time, versus running the calculation on a single CPU without that overhead.

Observation #4: *Execution Time* is linearly correlated with database size, as illustrated in Figure 12 (factor of 8 difference between the two lines; y-axis is in log scale). This helps to extrapolate how long a run might take based on database size. Additionally, this is an important element to the security aspect of the algorithm, because a non-linear runtime could reveal information to the DB Owner about the query and its similarity to the database entries.

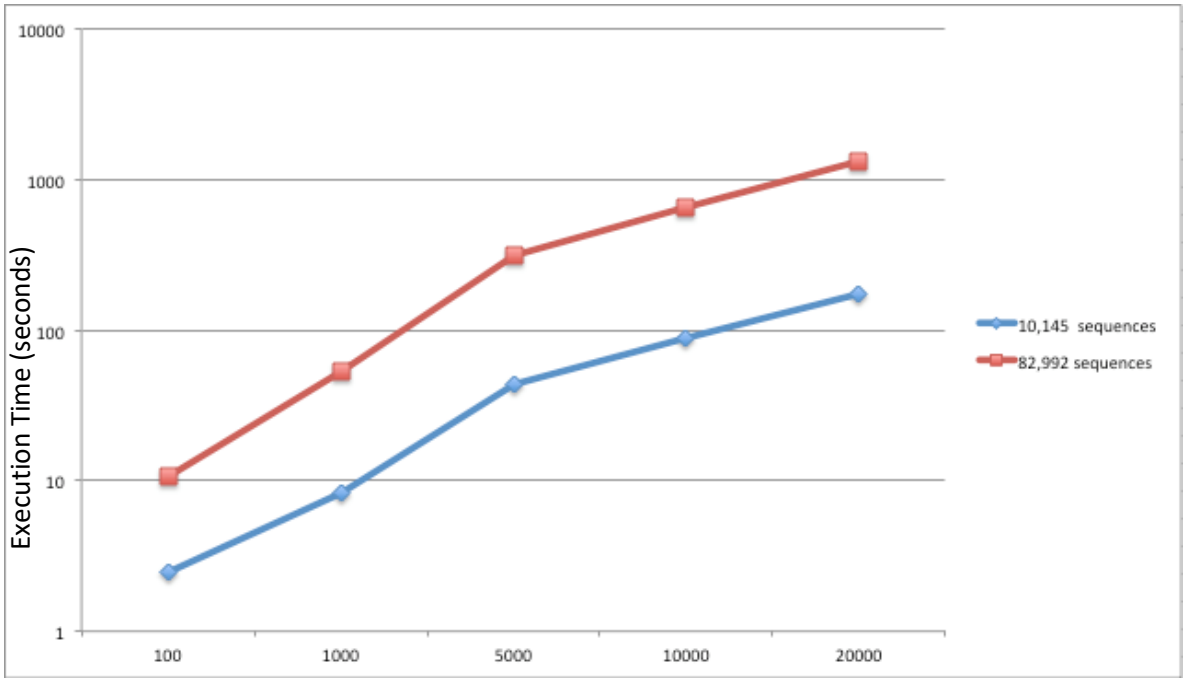

Figure 12: SIG-DB execution time for 100 and 1000 database entries compared to different query lengths; both runs performed with 48 cores. Note: y-axis is in log-scale.

Fully Homomorphic Encryption

In the above description, we employ Paillier homomorphic encryption (PHE), which allows for the use of addition on encrypted cipher text. A most robust version of this encryption is fully homomorphic encryption (FHE), which allows for the use of a larger set of operations on the encrypted cipher text, such as addition, division, and multiplication. These additional operations come with a computational overhead, making it more challenging to scale for most operational uses. There are companies, including Enveil (an IQT portfolio company), currently working to address this challenge.

Using FHE instead of PHE allows the algorithm to calculate the full IoU/IoQ/IoD scores within the database and return only these encrypted scores, instead of an encrypted intersection and plaintext LSH magnitude as done in the Paillier version of the algorithm. This reduces the information leaked about the database slightly by not revealing the LSH magnitudes.

The actual SIG-DB algorithm remains the same between FHE and PHE. Therefore, all performance measures except *practicality* will be the same for either PHE or FHE. To understand how significant the additional computational overhead would be for a FHE implementation of SIG-DB in its current form, we conducted queries with sequences of 1,000 base pairs against databases of 7, 70, 700, and 7,078 entries. We found that the FHE implementation requires approximately 2.25 times as long to execute as the PHE implementation (Figure 13). This a lower cost overhead

than we expected, as FHE is infamous for the time it takes to operate. This performance difference is most likely due to the programming language that each algorithm version is implemented in. The PHE SIG-DB is written in Python®, which is a high-level scripting language and is not optimized for complex mathematical operations without the use of additional packages. The FHE version of the SIG-DB algorithm employs a version of the Simple Encrypted Arithmetic Library (SEAL), created by Microsoft Research and written in C++. This programming language is much more efficient and is designed for fast and efficient operations. The FHE was implemented in a Python® binding version of SEAL that we developed in conjunction with Lab41, and subsequently released in an open-source repository named PySEAL.

The comparison of the PHE to the FHE version of the algorithms tells us that, even though the cost overhead was lower than expected (again likely due to implementation language), it is still substantial when using fully homomorphic encryption. Even written in a more efficient programming language, the FHE algorithm takes approximately 2.25 times as long to operate, and further work must be done, either in-house or in conjunction with our portfolio company, to improve this outcome.

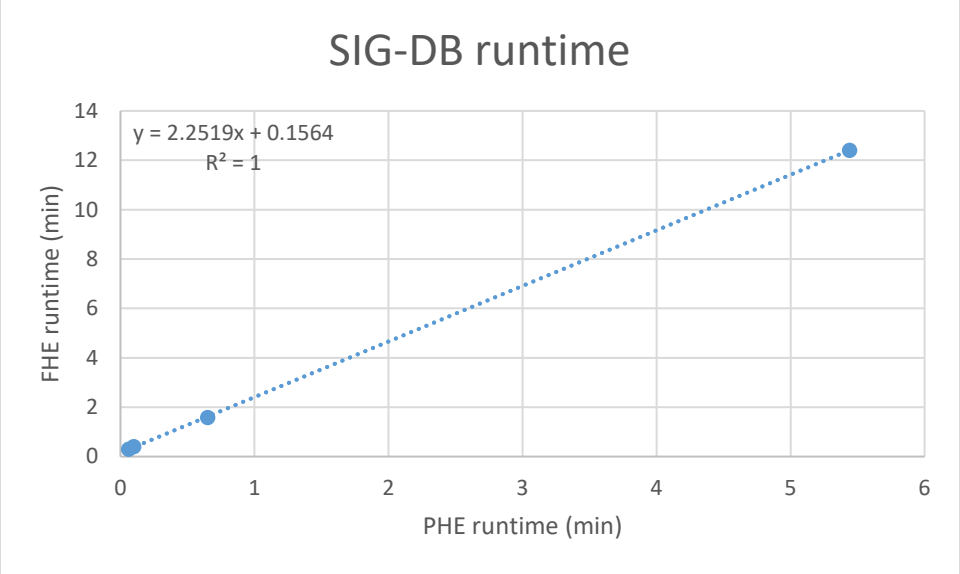

Figure 13: SIG-DB execution time for query sequences of 1,000 base pairs in length. The x-axis demonstrates partially homomorphic execution and the y-axis demonstrates fully homomorphic execution. The time point comparisons are for searching a data base of 7, 70, 700, and 7,078 sequences (ascending order of execution time).

Security Enhanced Third Party Approach

Using a trusted third party to execute a genomic comparison is an approach that could be used today. It involves a third party, trusted by both the Querier and the Database Owner, which executes any operations. Because the third party in this scenario receives both the database and the query, the level of trust required by the database and query owners is significant. However, advances in hardware-based secure computation could facilitate that trust. Intel® SGX offers an opportunity to enable a secure enclave in an otherwise unsecured operating system[33,34,35]; in our case, we envision a cloud-based system. This cloud-based set up (which we are referring to as a "Security-Enhanced Third Party") is not operationally available as of yet, although Microsoft Azure and IBM are actively working on this approach[36,37] (commercial release date TBD).

This Security-Enhanced Third Party approach would operate as follows:

- The Security-Enhanced Third Party system would issue the Querier and the Database Owner unique keys with which to encrypt their respective data.
- Both the Querier and the Database Owner would encrypt their data and upload to the cloud platform.
- The SIG-DB executable would be loaded into the SGX enclave in the cloud[38].
- The SGX enclave, having issued the encryption keys, would be able to decrypt both parties' data and perform a SIG-DB comparison.
- The resulting similarities would be re-encrypted using the encryption key provided to the enclave by the Querier (often referred to as "bring your own key" protocol) and the encrypted IoU/IoQ/IoD score would be passed back to the Querier
- The Querier would decrypt the similarity score and use the information in the manner best suited for the results (e.g. pursue further action or not)

The major benefit of secure enclave-based algorithms is the ability to operate on plaintext while maintaining privacy, and only needing to perform encrypt and decrypt operations on ciphertext. This would significantly speed up the similarity comparison stages of the SIG-DB algorithm. An additional benefit for the cloud-based system is that the Database Owner is no longer responsible for running the executable. This could address the concerns of some database owners about

[33] McKeen F et al, Innovative Instructions and Software Model for Isolated Execution. 2013. HASP. Accessed online 24 Feb 2018. https://docs.google.com/file/d/0B_wHUJwViKDaV0tsLUZnSWxCVnc/edit

[34] Anati I et al, Innovative Technology for CPU Based Attestation and Sealing. 2013. HASP. Accessed online 24 Feb 2018. https://docs.google.com/file/d/0B_wHUJwViKDaSUV6aUcxR0dPejg/edit

[35] Hoekstra M et al, Using Innovative Instructions to Create Trustworthy Software Solutions. 2013. HASP. Accessed online 24 Feb 2018. https://docs.google.com/file/d/0B_wHUJwViKDaNk5IWVBJeXRxc0k/edit

[36] Microsoft Azure. "Introducing Azure Confidential Computing" 14 Sept 2017. Accessed online 24 Feb 2018. https://azure.microsoft.com/en-us/blog/introducing-azure-confidential-computing/

[37] Schuster F et al, VC3: Trustworthy Data Analytics in the Cloud using SGX. 2016. Accessed online 24 Feb 2018. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/vc3-oakland2015.pdf

[38] The current implementation of SIG-DB allows for a user to specify plaintext-to-plaintext comparisons, which is what would be used in this scenario.

running someone else's executable behind their firewall. However, this system comes with limitations. The primary limitation of the secure enclave is that all processed data and software must be within the enclave. This requires the query and the data to be co-located on the same system at one point or another. If the query and the data are not already co-located, then this would require additional data transport time. Although the level of trust would likely be much lower than that of a traditional trusted third party, there still remains an element of trust that is needed, because the users must trust Intel® (the manufacturer of the SGX chip). Intel® has tried to address this by creating a key generation process that removes Intel® from knowing the actual key being used[39]. Moreover, the data and operation is protected from the host of the cloud by the data being encrypted when not in the secure enclave, and there is no need for the host to have access to any keys involved.

The complexity of the overall SGX system will likely cause performance degradation compared to plaintext operations on an open system – the expected amount is undetermined at this time given the limited amount of performance testing to date. One study[40] conducted by IBM Research found code running inside the enclave was able to reach high throughput for certain input sizes, on par with code running outside the enclave. However, they also found the entire SGX system to be very complex to performance tune with many slow-down points, including the context switching required to enter or exit the enclave. The number of steps contributing to the performance degradation depends on the actual set-up used with the enclave.

We originally set out to develop an SGX/SIG-DB approach using a local SGX-enabled CPU. We secured a laptop with the Intel® SGX chip and an SGX-supported motherboard. Using a Docker® image with Intel® SGX support[41] and the graphene library[42] (a library OS for Linux multi-process applications with SGX support that enables Python® code to operate within the enclave), we started developing the SGX-based algorithm. However, as noted below, we ran into four significant issues that ultimately halted our efforts.

First, developing with SGX code is complicated and requires a unique set of coding skills thatare not as common as Python®-based bioinformatics skills. We had SGX expertise in house, but only for the first part of the year, which supported our initial exploration efforts. As the SGX software ecosystem matures, the level of hardware and programming expertise may be reduced.

Second, the SGX environment is still immature. This is expected given that SGX was first announced in 2013, and it takes time to fully test and prove out any new hardware. We found the code to be buggy and its documentation minimal. . Recently, a new version of the graphene

[39] Aumasson JP and Merino L. SGX Secure Enclaves in Practice: Security and Crypto Review. 2016. Blackhat. Accessed online 14 Feb 2018. https://www.blackhat.com/docs/us-16/materials/us-16-Aumasson-SGX-Secure-Enclaves-In-Practice-Security-And-Crypto-Review.pdf

[40] Harnik D and Tsfadia E. "Impressions of Intel® SGX performance." 27 Dec 2017. Accessed online 24 Feb 2018. https://medium.com/@danny_harnik/impressions-of-intel-sgx-performance-22442093595a

[41] https://github.com/tozd/docker-sgx

[42] https://github.com/oscarlab/graphene

library was released that fixes many of the bugs we encountered, which would be used if we were to continue this development.

Third, a laptop is insufficient for the application we are pursuing. Most applications that are using SGX currently store data or keep ledgers (such as digital rights management, which spurred the development of SGX). However, our application requires the data to be operated on, which means the algorithm has to be stored and executed all within the secure enclave. The memory within that enclave on a laptop is too small to perform the level of operation we need, which is why we belive that, ultimately, only a cloud-based enclave system will have sufficient computational capacity.

Lastly, at mid-project, a significant security flaw was announced for the Intel® Management Engine that compromised the security of nearly all Intel® systems. Although Intel® was able to provide a quick fix through a firmware update, it is still undetermined if and how this affects the SGX system. Due to the challenges with development and unknowns around the newly revealed inherent security flaws in the system, we did not complete a full-scope algorithm implementation of a Security Enhanced Third Party Approach.

However, we still see promise in the technology, particularly for a cloud-based system. The SGX technology is also on the radar of IQT technical staff, who will track its progress for possible future investments. IQT Labs is negotiating access to the Microsoft Azure system. If we succeed, we intend to continue the development and testing of this Security-Enhanced Third Party Approach using the Microsoft Azure system when the SGX capability is implemented.

SIG-DB Security Assessment

The overall security of an encrypted data manipulation algorithm is fundamental to its success. Of the three security schemes we examined in this study, the quantity and type of information leakage decreases from PHE, to FHE, to SGX. The SIG-DB algorithm leverages established homomorphic encryption schemes to ensure that no outside (non-participating) party can learn the details of specific queries; only information about the query request patterns can be learned. It also ensures that no outside party can view the database records directly. Only metadata is exchanged in the form of encrypted LSH magnitudes, and thus individual data entries are never exposed outside the owners' own security systems. It is possible, however, for an outside party to infer the length of a database record sequence relative to other records through the database record LSH magnitude that is returned to the Querier. This information itself has low value, however, due to the possible diversity of types of sequences within a database and the inherent variability in sequence size amongst species.

From the metadata returned for both homomorphic encryption schemes tested, the Querier will learn the number of entries in the database. When combined with the computed magnitudes returned for the PHE approach and the similarity scores, the Querier could extrapolate the diversity within the database. Additionally, if the Querier repeated SIG-DB multiple times against

the same database using the same query sequence, the Querier could learn about any database modifications that may have occurred. This risk is limited, however, since the execution is controlled by the Database Owner, who retains the right to refuse repeat executions.

An FHE approach, such as Microsoft SEAL, reduces the amount of information leaked by not needing to reveal the magnitudes of Database LSHs (unlike PHE), and therefore does not reveal the length of the database entries. This is because FHE allows for division and multiplication, allowing the algorithm to calculate the full similarity score (while keeping it encrypted) before passing the information out of the database. However, FHE comes with an increased computational cost, as described above.

The SGX approach is theoretically the most secure of these three security schemes with the least amount of information leaked. This reduced information leakage is because the Database Owner is no longer receiving the query sequence and running the executable. The Querier could still learn some information about the database if a similarity score is returned for every database entry. However, a *max* function (which currently is not operationally feasible to implement within the SEAL FHE library) could be included in the operation so only the top N scores are returned; meaning very little (if any) information about the database could be extrapolated. For overall system security, the SGX system architecture is intended to lock down all information within the enclave. This means no component on the outside can penetrate the enclave, including kernels, hypervisors, SMM, BIOS, firmware, drivers, Intel® Management Engine, or any other part of the unsecured computing system. While theoretically secure, there still remain many questions about the actual operational security of the SGX system[43,44], including providing input to and output from the secure enclave without creating breaks in the security, vulnerabilities to attacks (e.g. side-channel attacks), exploitable bugs, and possible information leakages. These system securities will need to be addressed before SGX is widely accepted in the community. Researchers are working on answering these questions[45] and Intel® has provided open source libraries for developers to implement SGX and provide feedback on vulnerabilities of the code[46]. We remain fairly optimistic about the future of SGX and the capabilities it could be bring to bear once the technology has been sufficiently tested and optimized.

[43] Joanna Rutkowska. Thoughts on Intel's Upcoming Software Guard Extension, Part 1. The Invisible Things Lab's blog. 30 Aug 2013. Accessed online 24 Feb 2018. http://theinvisiblethings.blogspot.com/2013/08/thoughts-on-intels-upcoming-software.html

[44] Joanna Rutkowska. Thoughts on Intel's Upcoming Software Guard Extension, Part 2. The Invisible Things Lab's blog. 23 Sept 2013. Accessed online 24 Feb 2018. http://theinvisiblethings.blogspot.com/2013/09/thoughts-on-intels-upcoming-software.html

[45] Georgia Tech, School of Computer Science. College of Computing. "SGX Projects". Accessed 24 Feb 2018. https://sslab.gtisc.gatech.edu/pages/sgx.html

[46] Intel SGX for Linus. https://github.com/intel/linux-sgx

Conclusions and Next Steps

Through this cross-lab collaboration, we have demonstrated a suitable, technologically feasible approach to compare a genomic sequence of interest to a privately-held database, and produce an indication of similarity between the sequence of interest and each database entry. This algorithm simultaneously protects the security concerns of the Querier and the privacy concerns of the Database Owner. SIG-DB leverages established information security techniques, but does so in a way that is usable by non-experts. This approach could enable bioinformatics practitioners or investigators to leverage privately-held information in a secure way – a capability that is non-existent today for genomic sequence comparisons. This tool could expedite the timeliness of queries and gain meaningful results in critical timeframes for response decisions. This is important, as many bioinformatics practitioners are not security experts, and thus are less likely to develop applications in this space.

There are a variety of approaches one could take to enable secure computations, and they will each have their own strengths and weaknesses. Of the approaches we explored in this study, the SGX-enabled Security Enhanced Third Party approach offers the least information leakage (theoretically) and a minimal increased burden of time for the execution of the similarity comparison compared to plaintext, unsecure protocols (although, a full statement of comparison is not possible at this time since overall expected performance degradation is unknown). However, much research is still needed to fully measure and understand just how secure this system really is. The Paillier HE implementation of SIG-DB creates the greatest amount of information leakage of the explored approaches. However, this still remains low and is likely tolerable given the benefits such a system provides. While the Paillier HE implementation does add substantial computational time compared to plaintext operations, it is not as costly as the fully HE implementation, and is the most likely short-term solution for this application. The ultimate choice of which implementation to use comes down to the users' trade-off between tolerable execution time and information leakage.

B.Next hosted a roundtable event on February 6, 2018 to present and discuss SIG-DB with biodefense experts, bioinformatics practitioners, and analysts who are responsible for interrogating biological samples. The participants confirmed that data sharing is still a huge problem for national security purposes and will only continue to grow. There was unanimous interest in the SIG-DB approach, with recognition that the approach (and alternative approaches) should continue to be tested and optimized. A number of participants articulated how the ability to access private genome databases with such a tool would greatly enhance their ability to analyze new samples. In addition, they identified new use cases, including the searching of compartmented data within agencies, data searches between participants in commercial joint ventures, and leveraging protected genomic data for population studies in medical research. Attendees recommended a number of technical avenues for further investigation, including examining effect of k-mer size on performance and possible ways to incorporate the order of k-

mers into the analysis, exploring the use of Bloom filters, crafting a metric for data leakage, and exploring ways to speed up the execution time.

To decrease execution time and improve performance, possible approaches include:

- Characterizing additional scoring metrics to determine the optimal approach.
- Implementing the SIG-DB algorithm in C++ (which runs faster than code in Python®), and completing a pilot study with partners using such a C++ version to further evaluate practicality and real-world scenarios.
- Evaluating and optimizing the protocol for bacterial, plant, and animal genome sequences. (We limited the sequence lengths to 20,000 bases to enable faster runtimes during testing.)
- Exploring other secure computation techniques to identify the optimal approach based on performance and execution time.
- Evaluating the build and performance of SIG-DB with a Security Enhanced Third Party.

We developed this tool with the initial intention of it being used for microbial genomic sequence comparisons, especially when a user needs to quickly identify a source of information relevant to the query sequence, such as after the intentional release of a BW agent. However, there are a number of applications that it could be used for including human genomics, healthcare, organizational collaborations, and more. SIG-DB can be optimized for a particular application by modified the parameters (k-mer size, LSH size, hash collision rate) based on the tolerances of mutations, size of database, and computational resources. The potential versatility of this protocol creates an even greater opportunity for impact within the broader scientific community.

Appendix 1: Supplementary Materials

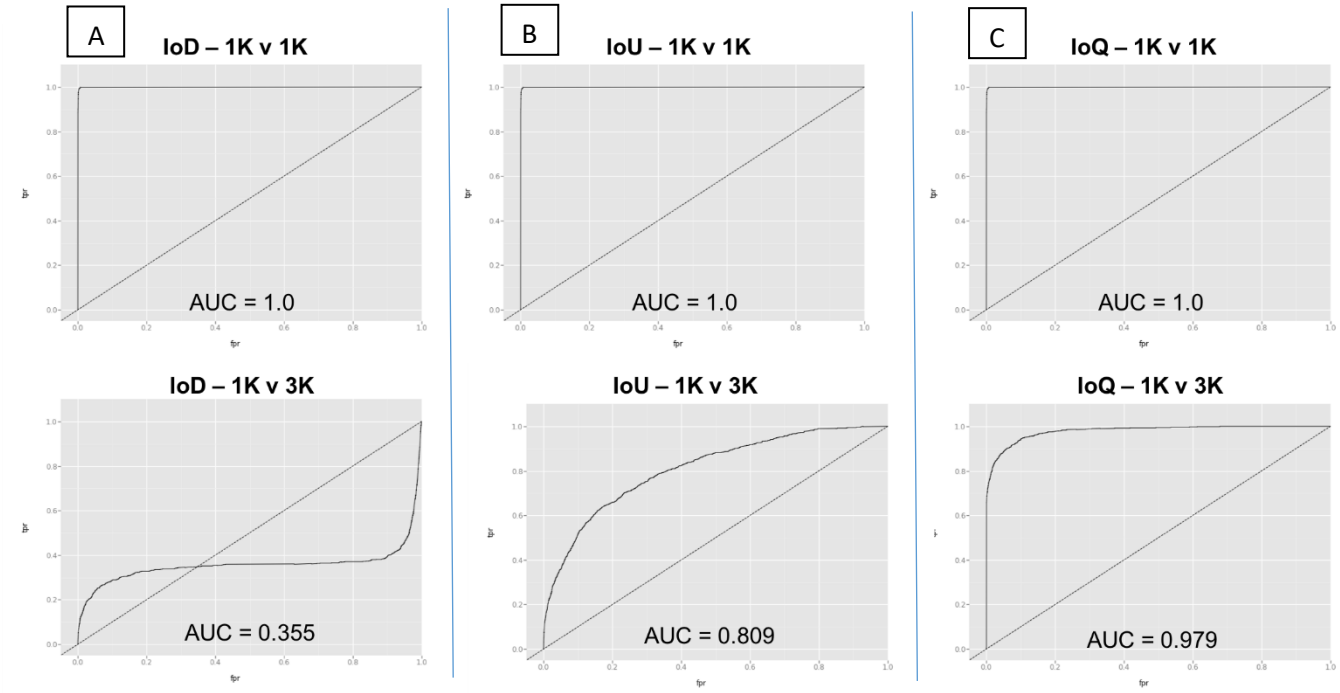

Figure 13: Receiver Operator Characteristic (ROC) curves for Paillier-SIG-DB based on each similarity score – (A) IoD, (B) IoU, and (C) IoQ. Query length =1000 bases and ANI threshold of 0.99. The DB entry length was 1000 bp (top row) with a total of 750,000 query to database entry comparisons and 1,515 of those with ANI scores above the ANI threshold (true positives) and 3000 bp (bottom row) with DB entry length=3000 bp with a total of 750,000 query to database entry comparisons and 1,708 of those with ANI scores above the ANI threshold.

As shown in Figure 13, using IoD as the similarity score when a query is one-third the size of the database entires results in an odd ROC curve representing very poor performance. One possible explanation is that the denominator is larger than the possible intersection space, and as such, we may be washing out the small differences between the intersections of positive and negative examples. In a dataset of only 0.2% positive samples, this leads to broad misclassification. This phenomenon is exactly why we included IoD, IoU, and IoQ as our scoring metrics. Operationally, the max similarity score should be used, as demonstrated by the near-perfect performance obtained in Figure 9.

Table 1. ANI Data statistics describing the distribution of the ANI scores for each database element length dataset.

Database element Length	Min	First Quartile	Mean	Third Quartile	Max	STD
1000	0.00	0.34	0.41	0.53	1.00	0.09
2000	0.00	0.36	0.42	0.51	1.00	0.12
3000	0.00	0.29	0.46	0.59	1.00	0.19