Leveraging Homomorphic Encryption to Securely Interrogate a Privately Held Microbial Genomic Database

Alexander J. Titus[1,2], Audrey Flower[3], Patrick Hagerty[4], Paul Gamble[3], Charlie Lewis[5], Todd Stavish[3], Kevin P. O'Connell[1], Stephanie M. Rogers[1]
[1]B.Next, IQT Lab, Arlington, VA; [2]Quantitative Biomedical Sciences, Dartmouth College, Hanover, NH; [3]Lab41, IQT Lab, Arlington, VA; [4]CosmiQ Works, IQT Lab, Arlington, VA; [5]Cyber Reboot, IQT Lab, Arlington, VA

Abstract

An effective response to a disease outbreak requires the rapid identification of pathogen and source.

Exploiting genomic information has become an important component of effective biothreat agent identification, characterization, and attribution. To do so, the necessary bioinformatic analyses require known genomic data against which to compare the agent's genomic data. However, more and more genomic data is becoming privately held.

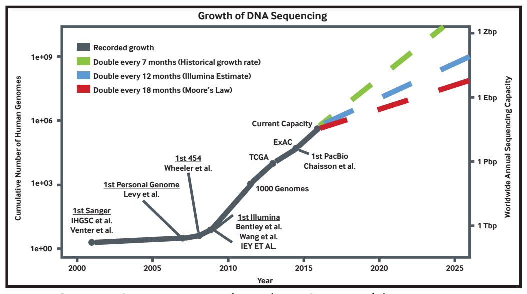
To truly understand where an agent came from or important features of the agent (e.g., virulence, alternative hosts, and environmental stability), the biodefense community will likely need to leverage the genomic data that resides in these private databases. This may be especially important when a truly novel agent is discovered and near-neighbors need to be identified.

Security requirements necessary for biothreat agent information or active investigations limit the direct sharing of genomic information with outside parties. Private entities are often unable to share access to their database due to privacy and legal issues. Fortunately, technology options exist that enable secure computations to be executed that fulfill data privacy requirements.

We developed the Secure Interrogation of Genomic Databases (SIG-DB) algorithm to enable the interrogation of a privately held database with a *sequence of interest* to determine the presence of similar sequences, without compromising the query or database information. This method was confirmed to be functional and evaluated using wild-type and in silico mutated versions of *Escherichia coli* and *Staphylococcus aureus* genomic sequences obtained from the NCBI RefSeq database[1].

Secure Genomic Comparisons

Genomic data is becoming increasingly valuable as methods emerge to use the information at scale, and as we develop a greater understanding of the genotype-phenotype relationship. Synthetic biology and the low cost of sequencing are increasing the amount of privately held genomic data. Data in storage can be protected by hardware or 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. Our results demonstrate one possible approach to enable the interrogation of a privately held database with a *sequence of interest* to determine the presence of similar sequences, without compromising the query or database information.



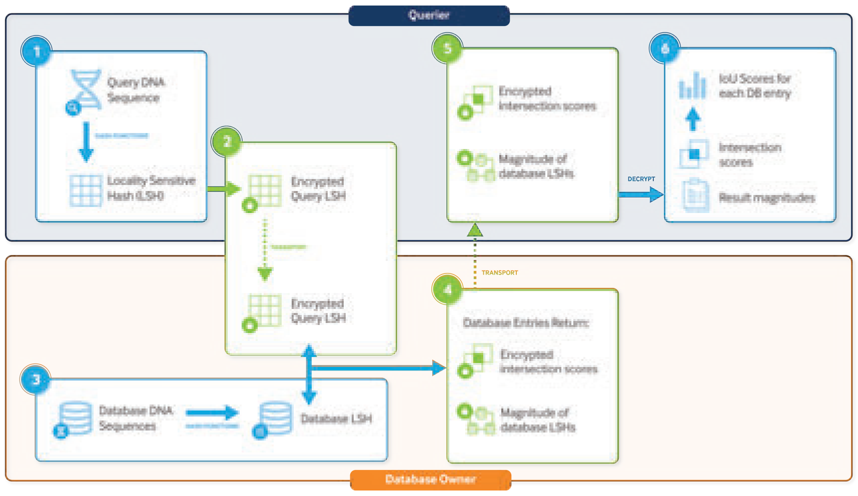
Source: Stephens et al. (2015) PLoS Biol 13(7): e1002195.

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 needed to respond to a biological event, whether natural or deliberate, by increasing the quantity and diversity of data used during critical bioinformatics analyses. Additionally, an approach like SIG-DB could enable government agencies to exploit data at other agencies or within their own agency that are 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. For example, disease-causing mutations in human genomes are being discovered and rely on population-level studies to identify causality. However, this data also holds our identity. So, protection and privacy is key.

The digital revolution transformed the way we live today, including how we protect and manage our data. The biorevolution is going to further transform the world, increasing the production and value of biological data. SIG-DB demonstrates how we can leverage existing data security approaches to continue to utilize genomic information that would otherwise be unreachable.

SIG-DB Protocol

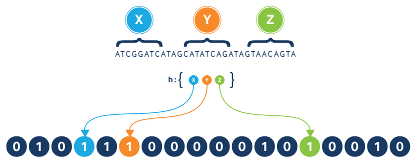

Querier
1 Query DNA Sequence
Locality Sensitive Hash (LSH)
2 Encrypted Query LSH
Encrypted Query LSH
3 Database DNA Sequences
Database LSH
4 Database Entries Return:
Encrypted intersection scores
Magnitude of database LSHs
TRANSPORT
5 Encrypted intersection scores
Magnitude of database LSHs
DECRYPT
6 IoU Scores for each DB entry
Intersection scores
Result magnitudes
Database Owner

The SIG-DB protocol is intended to be used between two "willing-but-unable" parties: Querier and Database Owner. The genomic data is split into k-mers and hashed into a locality-sensitive hash (LSH) to reduce data size and, therefore, compute time. SIG-DB leverages homomorphic encryption (an established, software-based data security method) to protect the query sequence. Encrypted similarity scores are generated for each comparison of the encrypted query LSH to every DB entry LSH, which are used by the Querier to determine if the database contains a sequence (or sequences) similar to their *sequence of interest*.

SIG-DB is available here: https://github.com/BNext-IQT/GEMstone

Locality-Sensitive Hash

A locality-sensitive hash (LSH) is a space efficient data structure that stores information as either a 1 or 0. The size of the LSH is preset but tunable, determined by the acceptable probability of overlapping hashes.



Similarity Scores

To account for sequence length variations, 3 metrics are used to calculate similarity: Intersection over Union (IoU), Intersection over length of Query (IoQ), and Intersection over length of Database entry (IoD).

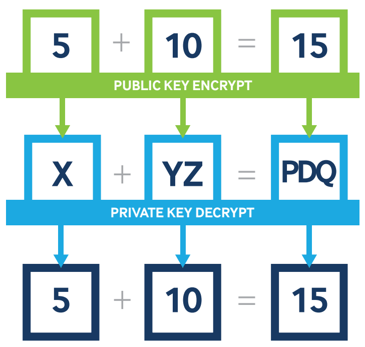
$$IoU = \frac{Intersection}{|Query\ LSH| + |DB\ entry\ LSH| - Intersection}$$

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

Homomorphic Encryption

Homomorphic encryption (HE) is a class of encryption schemes that allow mathematical operations to be carried out on the encrypted text in the same manner as operations carried out on the plain text.

Partially HE schemes allow a subset of operations to be performed, such as addition, while fully HE schemes[2] allow for an arbitrary set of operations, such as addition and multiplication, to be performed.



The current SIG-DB implementation uses Paillier additive HE[3].

PySEAL

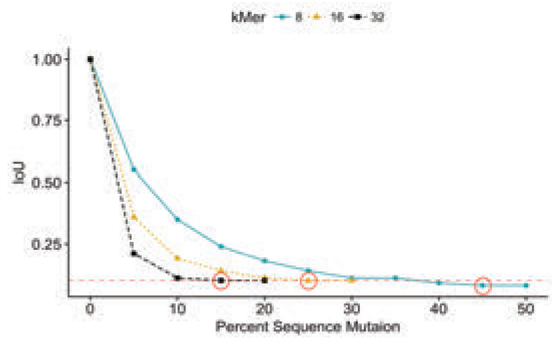
Python-based fully HE library for bioinformatics

HE is a promising application for secure computation in biomedical research and clinical data applications. The current SIG-DB implementation uses the Paillier HE scheme (PHE), which provides an adequate level of data privacy and protection for the current application. However, if additional privacy and protection is needed, a fully HE approach could be utilized, but it comes at a computational expense greater than PHE.

The Simple Encrypted Arithmetic Library (SEAL[4]) HE library is a well-established and maintained fully HE implementation. However, the software is provided in C++. We developed a Python wrapper for the SEAL library to provide the bioinformatics community with a tool to develop new methods using homomorphic encryption. The tool can be found here:

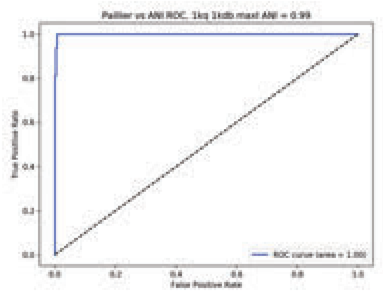
https://github.com/Lab41/PySEAL

Parameter Testing



Using 50 *E.coli* genomes and 50 *S. aureus* genomes, SIG-DB algorithm performance was tested at varying proportions of sequence mutations for three different k-mer sizes (k = {8, 16, 32}). **Success is defined by the algorithm returning the matching database element as the highest IoU in the database.** Red circles indicate the mutation limit for each k-mer size.

ROC Curve



To assess the ability of SIG-DB to correctly score the database entries against the sequence of interest, we compared the IoU score to average nucleotide identity (ANI: 'true positive' ≥ 0.99).

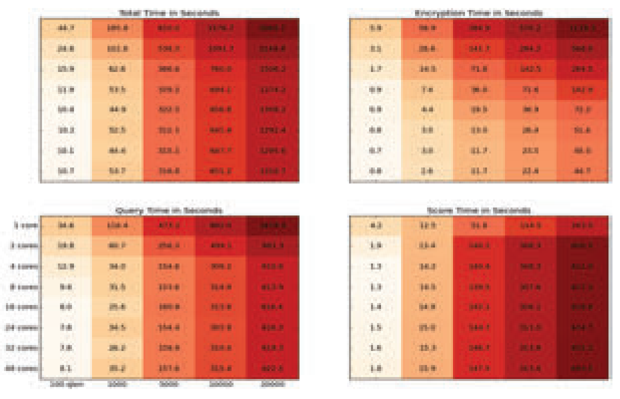
To generate the ROC curve, tests were executed with query and database sequences of equal length (1000 base pairs).

In this test, SIG-DB achieves an AUC of 1.00 when classifying a sequence from a database as "similar" to the query sequence.

Computation Time

Runtime of the SIG-DB algorithm is dependent on three steps: (1) encryption, (2) query, and (3) scoring.

Tests were executed to assess the run time of the algorithm. Our results demonstrate that performance gains are possible using an increased number of parallel cores.



The current implementation requires ~32 hours to execute against a DB of 1.5GB.

References:
[1]NCBI viral genome resource. Nuc. Acids. Red. 2015
[2]A fully homomorphic encryption scheme. Stanford U. 2009
[3]Public-key cryptosystems based on composite degree residuosity classes. Int. Conf. on Theory and Apps. Of Crypto. Techs. 1999
[4]Simple Encrypted Arithmetic Library. Microsoft Research. 2018