Secure Genomic Computation through Site-Wise Encryption

Yongan Zhao XiaoFeng Wang[2] Haixu Tang[1]

School of Informatics and Computing, Indiana University, Bloomington, IN 47405

[1]: primary advisor; [2]:co-advisor

Abstract

Commercial clouds provide on-demand IT services for big-data analysis, which have become an attractive option for users who have no access to comparable infrastructure. However, utilizing these services for human genome analysis is highly risky, as human genomic data contains identifiable information of human individuals and their disease susceptibility. Therefore, currently, no computation on personal human genomic data is conducted on public clouds. To address this issue, here we present a site-wise encryption approach to encrypt whole human genome sequences, which can be subject to secure searching of genomic signatures on public clouds. We implemented this method within the Hadoop framework, and tested it on the case of searching disease markers retrieved from the *ClinVar database against patients' genomic sequences*. The secure search runs only one order of magnitude slower than the simple search without encryption, indicating our method is ready to be used for secure genomic computation on public clouds.

Background

With the advance of next-generation sequencing (NGS) technology, massive human genomic data (from whole-genome sequencing or exome sequencing) has been accumulated rapidly in laboratories and clinical settings [1]. The availability of these data accelerates the exploitation of the field of personal medicine and genome-based healthcare. To efficiently explore these datasets, research institutions and healthcare practitioners need to build infrastructure (i.e., computer systems with large memory and disk space, powerful CPUs and fast network connections) to support intensive genome computation. In addition to the high maintenance cost, human genome data comes with high liability: it contains identifiable information of human individuals (e.g., disease patients), and thus has to be protected from un-authorized access. As a result, computer servers have to be dedicated to the genome data from each specific project (or group of patients), which increases the administrative cost of the data analysis.

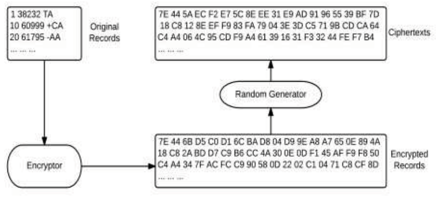
Commercial (also referred to as the public) clouds (such as the Amazon EC2 and the Microsoft Azure) have already been widely utilized in many fields to deliver the on-demand computation resources with pay-as-you-go pricing. Adopting an elastic computing model, commercial clouds consist of commodity machines and provide redundant data storage and high parallelism of computation capability. The cost-effectiveness of commercial clouds has been well recognized in bioinformatics for processing massive genomic data [2]. Currently, many tools are available on commercial clouds [3], for instances, reads mapping, fragment assembly and function analysis of the genomic data from microbial organisms, animals and plants. However, utilizing commercial clouds for analyzing sensitive human genomic data is hindered by the privacy concerns [4-7]. On the one hand, it is well known that even a small piece of genomic information (e.g., the genotypes over a few dozens of SNPs) can be used to infer the identity of an individual or the potential disease risk. Statistical methods [8-12] were also developed to infer the presence of a participant in a case group from the group's aggregate genomic data (e.g., allele counts on the SNP sites across the whole genome). On the other hand, commercial clouds do not offer high security assurance and tend to avoid any liability [13, 14] because in the elastic computing model, the physical servers on the cloud are shared by many users. Due to such conflict, the analysis of personal genome datasets is rarely outsourced to commercial clouds.

One can address the privacy concerns by encrypting the sensitive human genomic data before handing them over to commercial clouds. As used for storing private databases such as bank accounts on public clouds, the encrypted data provides high security assurance. There are growing interests in developing efficient protocols to support computation on encrypted data stored based on different adversary models and computing assumptions, including secure multi-party computation (SMC), oblivious RAMs, homomorphic encryption, functional encryption, property-preserving encryption, and searchable symmetric encryption (for a review see [7]).

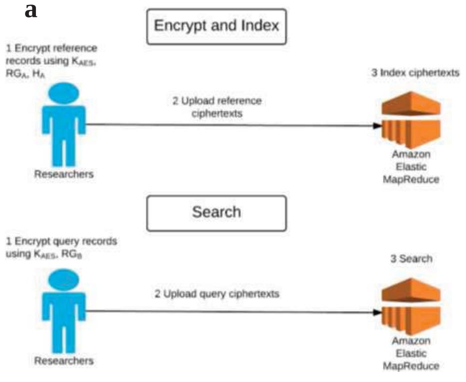
In this paper, we present a simple cryptographic approach that supports the search of genomic signatures (e.g., one or more single nucleotide variations (SNVs) associated with a certain disease) on commercial clouds through the site-wise encryption (SWE) of whole human genomic sequences. In our adversary model, commercial clouds are reasonably assumed to be honest but curious, i.e., they are assumed to return the results for any submitted computing jobs while ensuring the correctness and integrity of results. Meanwhile, they may try to learn some statistics, such as access patterns, that can be used to infer private information from the submitted data. We also assume that the data storage can be compromised, but the computation procedure itself is protected (i.e., we do not intend to protect the computing results). Finally, we implemented our approach on the Hadoop framework; but this should not be a constraint for our approach since it can be easily extended into other platforms.

Secure Search of Genomic Signatures

We study the following genomic signature search problem: given the whole genome sequences of an individual, and a set of SNVs known to be associated with a phenotype, we want to know if the individual's genome has these SNVs (or the disease susceptibility of the individual). Our goal is to develop a secure computing protocol to solve this problem on commercial clouds, while the individual's genomic sequences are protected. More specifically, what we want to achieve are as follows: (1) public clouds do not know the content of the data, though they see how frequent each record has been queried; (2) an attacker, even when observing the queries from the user (through network eavesdropping) and obtaining encrypted SNVs (e.g., by temporarily compromising cloud storages), knows nothing about the content of the queries, SNVs and even the frequencies of the queries. We note that such a solution could be applied to several practical scenarios in genome-based medicine that require large computing resources, for examples, 1) to screen the risks of genetic diseases by searching the whole genome of an individual against the database of genetic diseases (e.g., ClinVar [15]); 2) to identify potential pharmacogenomic markers (e.g., in PharmGKB database [16]) relevant to drug dosage or selection of effective treatment; and 3) to classify patients with indistinguishable symptoms into disease subtypes based on genetic markers [17].


1 38232 TA
10 60999 +CA
20 61795 -AA
... ...
Original Records
7E 44 5A EC F2 E7 5C 8E EE 31 E9 AD 91 96 55 39 BF 7D
18 C8 12 8E EF F9 83 FA 79 04 3E 3D C5 71 9B CD CA 64
C4 A4 06 4C 95 CD F9 A4 61 39 16 31 F3 32 44 FE F7 B4
... ...
Ciphertexts
Random Generator
Encryptor
7E 44 6B D5 C0 D1 6C BA D8 04 D9 9E A8 A7 65 0E 89 4A
18 C8 2A BD D7 C9 B6 CC 4A 30 0E 0D F1 45 AF F9 F8 50
C4 A4 34 7F AC FC C9 90 58 0D 22 02 C1 04 71 C8 CF 8D
... ...
Encrypted Records

a


Encrypt and Index
1 Encrypt reference records using K_AES, RG_A, H_A
2 Upload reference ciphertexts
3 Index ciphertexts
Researchers
Amazon Elastic MapReduce
Search
1 Encrypt query records using K_AES, RG_B
2 Upload query ciphertexts
3 Search
Researchers
Amazon Elastic MapReduce

b

Figure 1. The cryptographic protocol for genomic signature search consisting two steps: (a) the site-wise encryption of the whole genome sequences; and (b) the secure search on public clouds.

To encrypt whole genome sequences, we first compare it against the reference human genome, and encode it as a set of variations between them, including the single nucleotide variations (substitutions or insertion/delections), long insertions/deletions, micro-inversions, and translocations. Notably, because the number of variation sites (typically several millions of them) is relatively small comparing to the whole genome sequences (3 billions of bases), this conversion effectively reduces input data size while retaining all individual genomic information, and thus is often used to compress the whole genome sequences[18]. The collection of variation sites can be encrypted effectively in a site-wise fashion: each variation site can be encrypted separately using the same encryption key. Figure 1a illustrates the encryption procedure. The data owner generates a random key, a random string generator and a hash function, and encrypts each record (including a variation, its genomic location and other information) on a local machine (see **Methods** for detailed description of the encryption protocol). The resulting ciphertexts are then uploaded to commercial clouds. When the data owner wants to search a genomic signature (e.g., one or more single nucleotide variations at specific genomic locations), he needs to encrypt the signature in the same site-wise fashion (i.e., to encrypt each variation site separately) using the same key. As a result, the same variation in the whole genome sequences will lead to the same encrypted record, although the variation itself (the type or the genomic location) cannot be inferred from it. After the encrypted genomic signature is uploaded to public clouds, a simple string matching can be conducted to determine if each variation in the genomic signature is present or not in the whole genome sequences of interest (Fig 1b). This process does not leak any information about the original records, as only encrypted data is accessed. Moreover, each variation site is unique (i.e., at a different genomic location), and thus each encrypted record is non-redundant, which prevent the inference attacks using frequency analysis on encrypted data [19].

Methods

We devised two secure search (referred to as the basic and the randomize-verify) schemes based on the following security primitives.

A symmetric key encryption function (E). A symmetric key encryption function is a family of encryption functions, which uses one same key to perform both encryption and decryption. We adopt symmetric key encryption protocol, as in our scheme, it is the data owner himself that will conduct the search on public clouds. Due to its well-designed security attributes and efficiency, AES algorithm is used in our implementation, which is a block cipher and encrypt a fixed block (128 bits) in each operation by permutation and substitution.

A cryptographic pseudorandom generator (RD). We use it to generate non-deterministic seeds and strings in our second scheme to randomize the encrypted records (Fig 1a), which further strengthen the security assurance.

A cryptographic hash function (H). A cryptographic hash function digests records and produces hash values in a collision resistant way. It is practically impossible to invert hash values to original messages. We use it in the verification phase to determine if a pair of reference and query ciphertexts is from the same original record. SHA-256 algorithm is used in our implementation.

There are two security parameters (n and m) used in our schemes. Either n or m should be less than the length of an encrypted record, while their sum should be equal to the length of an encrypted record.

Basic scheme

In the basic scheme, the data owner first generates a random key (K_{AES}) for all records (variations) in the input genome (referred to as the reference records), and encrypts each record using E and K_{AES}. The encrypted reference records are then uploaded to public clouds after shuffling. This step needs to be done only once for a particular genome as the encrypted records can be kept on public clouds for future searches. If the data owner wants to search a genomic signature (i.e., a small set of variations), he will encrypt them using the E and K_{AES} of the corresponding genome to be searched against, and upload the encrypted records (referred to as the query records) to the cloud. Within the Map-Reduce framework (e.g., Hadoop), we use a random sampler to split the reference records into different reducers to balance the workload on every reducer. The reference records in each reducer are then indexed as a dictionary to be searched by using a binary search algorithm. When the Map-Reduce system starts to look for the exact matches between reference and query records, it automatically, in the shuffling phase, sorts query records and deliver each of them to a specific reducer, in which a binary search is initiated to determine if an exact match can be found for each query record. In the end, the match (or no match) of each query record will bel reported.

Randomize-Verify (RV) scheme

A problem of the basic scheme is that an attacker who monitors queries observes the frequencies of different queries, which could be an information leak of concern. Figure 2 illustrates an enhanced approach capable of withstanding this threat, called randomize-verify scheme. The idea is to make the ciphertexts for the different instances of the same query look different, thereby thwarting the attempt to accumulate the frequency of a specific query. Specifically, to encrypt a reference record g (Fig 2a), the data owner generates a n-bit random string (rs_R) by using RD, in addition to the key K_{AES}. He first applies E with the key K_{AES} to g, resulting in the encrypted record $E_{K_{AES}}(g)$, and then computes the cyphertext $C_R(g) = E_{K_{AES}}(g) \oplus (rs_R \cdot H_{0:m}(rs_R))$ by using the XOR operation between the encrypted record and the n-bit random string concatenated with its first m-bit hash value ($H_{0:m}(rs_R)$) computed by the hash function H, where $\cdot$ denote the concatenation operation on strings.

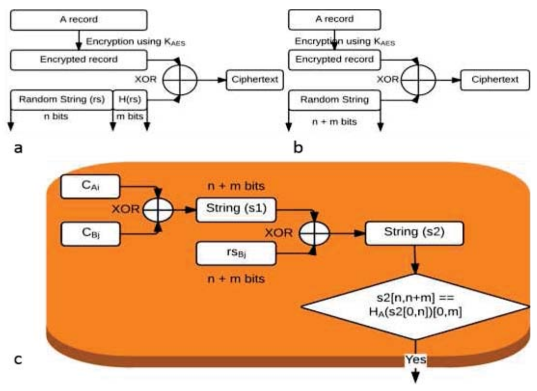


Figure 2. The randomize-verification scheme has three steps: (a) reference records encryption; (b) query records encryption; and (c) search on public clouds.

After ciphertexts are shuffled and uploaded to public clouds, they are first indexed to achieve the best performance of searching. The first 2-byte of an encrypted record ($H_{0:16}(E_{K_{AES}}(g))$) is used as a classification identifier. A random sampler is used to sample the ciphertexts to decide to which reducer a ciphertext is delivered to balance the workload of each reducer. These split databases are serialized on the local disk of each reducer so that every time when a new search task initializes, they are able to be loaded into the memory at the local machines where reducers are executed.

For the search of a genomic signature, each query record is encrypted in a similar way (Fig 2b). First, a common seed (S_Q) is generated for this specific search and a random seed (S) is generated for each query record (g'). The combination of both S_Q and S is used as the random seed to generate a $(n+m)$–bit random string ($rs_Q = RD(S_Q, \mathrm{S})$) for the query record g'. The ciphertext ($C_Q(g')$) is then computed by the XOR operation between the encrypted record $E_{K_{AES}}(g')$ (using E with the same K_{AES}) and the $(n+m)$–bit random string: $C_Q(g) = E_{K_{AES}}(g') \oplus rs_Q$. Finally, both $C_Q(g')$ and the corresponding S are uploaded to public clouds. The common seed can be transferred to the cloud through a secure channel (e.g., the SSH connection).

Finally, the secure search of the genomic signature can be conducted on public clouds by using a verification protocol (Fig. 2c). Every pair of reference and query ciphertext is verified (by the reducer on which the ciphertext share the same classification identifier) for representing the same variation or not by checking if $H_{0:m}(s2_{0:n}) = s2_{n:n+m}$ is true. The correctness this verification can be easily proved:

1) $s1 = C_R(g) \oplus C_Q(g) = E_{K_{AES}}(g) \oplus \left(rs_R \cdot H_{0:m}(rs_R)\right) \oplus E_{K_{AES}}(g) \oplus rs_Q = \left(rs_R \cdot H_{0:m}(rs_R)\right) \oplus rs_Q$
2) $s2 = \left(rs_R \cdot H_{0:m}(rs_R)\right) \oplus rs_Q \oplus RD\left(S_Q, \mathrm{S}\right) = rs_R \cdot H_{0:m}(rs_R)$

so $H_{0:m}(s2_{0:n}) = s2_{n:n+m}$ if and only if $g = g'$.

Results

We implemented our schemes in Java within the Hadoop framework. It supports inputs in both plain text format (shown in Fig 1a) and variant call format (VCF). The source code of the project can be found at http://swecloud.sourceforge.net. For the testing, we used the genome sequences from a participant of Personal Genome Project (PGP) [20], from which a total of 4,005,829 variation records were retrieved. We selected 45 disease associated SNVs (i.e., query records) in ClinVar [15] to test the secure search algorithm. A full list of these SNVs and their associated diseases can be found on http://omics.informatics.indiana.edu/mg/SWECloud/, e.g., T→A substitution at 51,078,333 on chromosome 18 (ClinVar ID: rcv000021740).

In order to estimate the computation overhead of the secure search schemes, we also implemented a simple binary search algorithm to identify query records in the genome sequences in Java, and ran it on a local single CPU machine (referred to as the plain search).

Table 1 shows the performance of two secure search (basic and RV) schemes in comparison with the plain search. In the encrypt phase, the RV scheme takes longer time on reference record encryption than the basic scheme (note that this is a one-time computation), as it needs to randomize data to achieve higher security guarantee (see Discussion). The encryption for query records takes negligible time. In both of the indexing and searching phases, these two schemes show similar performances. Because the RV scheme compares each pair of query and reference ciphertexts with the same 2-byte classification identifier, the basic scheme may show slightly better performance when we have more query records. Both schemes take more time in indexing and searching comparing with the plain search. But the computation overhead is not high, at approximately one order of magnitude, which can be compensated easily with the larger computing resources available at commercial clouds.

Table 1. The performances of implemented schemes.

	Encryption (seconds)		Indexing (seconds)			Searching (seconds)		
	Reference	Query	Mapper	Reducer	Total	Mapper	Reducer	Total
Basic	37.1	0.2	59.0 (2) *	109.2 (4) *	49.0	4.6 (1)	76.5 (4) *	21.3
RV	63.7	0.2	59.4 (2)*	84.4 (4) *	46.2	4.6 (1)	75.6 (4) *	21.5
Plain	-		7.3			6.1		

(*: The numbers in parenthesis represent the numbers of parallel mapper/reducer jobs. Total running time summed over all jobs are reported.)

Discussion

The privacy risks of genome computing on public clouds are two-folded: the data stored on public clouds can be used to infer sensitive information while at each time of computation, the query data may also be identifiable. The first risks can be completely mitigated in both schemes, as the security of the encrypted records is assured by the cryptographic primitives. An adversary cannot infer useful information without knowing the encryption key even if he obtains the ciphertexts. He cannot conduct frequency analysis on the encrypted reference records either because each variation appear at most once in the genome of any single individual while the genomes of different individuals will be encrypted using different key. On the other hand, an adversary may accumulate all query ciphertexts within a period of time and then try to infer sensitive information, such as potential disease susceptibility, based on those query ciphertexts and other public information. The basic scheme may be subject to this attack. As a result, the data owners may need to change the encryption key periodically, and re-encrypt the reference records so that the adversary cannot collect the sufficient query ciphertexts to analyze the pattern. Our second scheme further decreases these risks. The reference records are double protected, by the encryption key as well as the random strings that are independent on the original records. These random strings not only protect the encrypted records, but also make ciphertexts indistinguishable. It also prevents the pattern analysis of the query ciphertexts: even if an adversary accumulates all query ciphertexts, they cannot determine if two ciphertexts in different queries are the same.

Our schemes provide a secure yet efficient way of analyzing sensitive genomic data on public clouds, based on which one can outsource not only the sensitive computation but also the sensitive genome data to clouds. The data owners (e.g., healthcare practitioners) can store encrypted genome data from all individuals (e.g., patients) on public clouds, while only keeping the encryption keys (one for each individual) locally. As such, the data owner has low liability on a large amount of sensitive data, whereas the genome data can be analyzed on clouds whenever needed.

Many schemes have been proposed for secure genome computation, most of which are based on relatively expensive cryptographic primitives (typically with >4 orders of magnitudes of computation overhead), such as secure multi-party computation (SMC) and homomorphic encryption (HE) [21, 22]. Despite their high security assurance, our schemes based on site-wise encryption (SWE) of genomic variations are designed specifically for the secure search of human genomic data, and thus are simple and more efficient than generic schemes such as SMC and HE. Our schemes are ready to be used in real-world human genome computation, such as the genome signature search.

By adjusting n and m, we can assure the space of random strings is significantly larger than the number of encrypted records. In our implementation, we choose $n = 80$ and $m = 48$, as a person has about 4 millions of variation records in practice (thus $2^{80} \gg$ the maximum number of variation records of a person).

Acknowledgements. This work is supported by NHGRI/NIH (1R01HG007078-01) and NSF (CNS-1408874).

References

1. Ohno-Machado L. Sharing data for the public good and protecting individual privacy: informatics solutions to combine different goals. Journal of the American Medical Informatics Association. 2013;20(1):1-.
2. Stein LD. The case for cloud computing in genome informatics. Genome Biol. 2010;11(5):207.
3. Forer L, Lipic T, Schonherr S, et al. Delivering bioinformatics MapReduce applications in the cloud. Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on; 2014: IEEE.
4. Shoenbill K, Fost N, Tachinardi U, Mendonca EA. Genetic data and electronic health records: a discussion of ethical, logistical and technological considerations. Journal of the American Medical Informatics Association. 2014;21(1):171-80.
5. Lin Z, Owen AB, Altman RB. Genomic research and human subject privacy. SCIENCE-NEW YORK THEN WASHINGTON-. 2004:183-.
6. Erlich Y, Narayanan A. Routes for breaching and protecting genetic privacy. Nature Reviews Genetics. 2014;15(6):409-21.
7. Naveed M, Ayday E, Clayton EW, et al. Privacy and Security in the Genomic Era. arXiv preprint arXiv:14051891. 2014.
8. Gymrek M, McGuire AL, Golan D, Halperin E, Erlich Y. Identifying personal genomes by surname inference. Science. 2013;339(6117):321-4.
9. Homer N, Szelinger S, Redman M, et al. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. PLoS genetics. 2008;4(8):e1000167.
10. Craig DW, Goor RM, Wang Z, et al. Assessing and managing risk when sharing aggregate genetic variant data. Nature Reviews Genetics. 2011;12(10):730-6.
11. Sankararaman S, Obozinski G, Jordan MI, Halperin E. Genomic privacy and limits of individual detection in a pool. Nature genetics. 2009;41(9):965-7.
12. Humbert M, Ayday E, Hubaux J-P, Telenti A. Addressing the concerns of the lacks family: Quantification of kin genomic privacy. Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security; 2013: ACM.
13. Armbrust M, Fox A, Griffith R, et al. A view of cloud computing. Commun ACM. 2010;53(4):50-8.
14. Ristenpart T, Tromer E, Shacham H, Savage S. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. Proceedings of the 16th ACM conference on Computer and communications security; Chicago, Illinois, USA. 1653687: ACM; 2009. p. 199-212.
15. Landrum MJ, Lee JM, Riley GR, et al. ClinVar: public archive of relationships among sequence variation and human phenotype. Nucleic acids research. 2013:gkt1113.
16. Whirl-Carrillo M, McDonagh E, Hebert J, et al. Pharmacogenomics knowledge for personalized medicine. Clinical Pharmacology & Therapeutics. 2012;92(4):414-7.
17. Bioulac ‐ Sage P, Rebouissou S, Thomas C, et al. Hepatocellular adenoma subtype classification using molecular markers and immunohistochemistry. Hepatology. 2007;46(3):740-8.
18. Christley S, Lu Y, Li C, Xie X. Human genomes as email attachments. Bioinformatics. 2009;25(2):274-5.
19. Zerr S, Olmedilla D, Nejdl W, Siberski W. Zerber+ r: Top-k retrieval from a confidential index. Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology; 2009: ACM.
20. Church GM. The personal genome project. Molecular Systems Biology. 2005;1(1).
21. Baldi P, Baronio R, De Cristofaro E, Gasti P, Tsudik G, editors. Countering gattaca: efficient and secure testing of fully-sequenced human genomes. Proceedings of the 18th ACM conference on Computer and communications security; 2011: ACM.
22. Ayday E, Raisaro JL, Hubaux J-P, Rougemont J. Protecting and evaluating genomic privacy in medical tests and personalized medicine. Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society; 2013: ACM.