2015 IEEE 28th International Symposium on Computer-Based Medical Systems

Privacy-Aware Genome Mining: Server-Assisted Protocols for Private Set Intersection and Pattern Matching

Constantinos Patsakis, Athanasios Zigomitros
Department of Informatics, University of Piraeus
Piraeus, Greece
{kpatsak,azigomit}@unipi.gr

Agusti Solanas
Smart Health Research Group
Dept. Computer Engineering & Mathematics, Universitat Rovira i Virgili
Tarragona. Spain
agusti.solanas@urv.cat

***Abstract*—The Human Genome Project has generated a great wealth of information. Currently, almost all human genome has been sequenced and now it is time to identify the functionality of each gene. The sequence of base pairs accounts for approximately 3 billion elements. While there are many efficient algorithms and implementations to mine this information, doing it privately is a great challenge. Current state-of-the-art methods have improved their efficiency, but they are not practical yet. In this article, we introduce several protocols to drastically boost the performance of genome mining processes while guaranteeing privacy, thus, enabling practical implementations. We describe how to solve the *private set intersection* problem and a set of pattern matching queries with privacy. The proposed protocols are server-assisted and we prove that they are secure under the semi-honest model. We report the assessment of our solution using synthetic datasets and prove their efficiency.**

***Index Terms*—Privacy; secure multi-party computation; Private Set Intersection; DNA mining**

I. Introduction

The discovery of the three-dimensional double helix of DNA by Watson and Crick in the 50s revolutionized modern biology and led to astonishing results and discoveries. The Human Genome Project[1] managed to map and sequence all of our genes in 2003, enabling us to read the *instructions* about how human beings are created. These DNA instructions are written with an apparently simple four-letters alphabet of nucleotides, namely adenine, cytosine, guanine and thymine (A, C, G, T), and the length of this blueprint is approximately 3 billion *letters* long. Based on these instructions, researchers are struggling to map these abstract *rules* into functionalities, in order to understand how human beings are actually working and evolving, and how to detect patterns that make some individuals prone to specific diseases.

The continuous study of the human genome and the efforts to improve the procedures of DNA sequencing have led to a remarkable cost reduction, so that the cost for individuals is less than $100. Simultaneously, many organizations allow individuals to publish their DNA sequences, or parts of them, in order to enrich the current datasets and facilitate research focussed on the early diagnose and treatment of diseases. This wealth of data is drastically improving the research in this field that lead to better results. However, these improvements are not without a cost. When someone publishes his/her genome, it is possible to infer many things about her and her relatives. Moreover, since our genome can reveal disease predisposition [1], physical attributes [2], or even political beliefs [3], very sensitive personal information could be disclosed.

In any case, it should be noted that our DNA sequences are very stable[2], and the study of human beings DNA sequences is progressively producing new results. Therefore, by publishing and sharing our genome sequences, we do not know to what risks will we be exposed to in the near future, *e.g.* genome discrimination (eugenics) as portrayed in many sci-fi novels and movies. Obviously, to avert possible present and future problems and ethical and privacy-related issues [4][5] associated to the open sharing of DNA sequences, there is a need for enabling the analysis and data mining of such sequences with privacy-aware mechanisms.

Privacy-aware data mining of genome sequences is a very wide problem. In this article, we follow the same trail of thought than Dong et al. [6] and Kamara et al. [7], and propose several server-assisted protocols to further improve the performance of data mining processes over genome sequences. More precisely, the protocols that we propose can efficiently answer the following questions:

(Q1) Given $A, B \in \{A, C, T, G\}^m$ determine $A \cap B$.
(Q2) Given $A, B \in \{A, C, T, G\}^m$ determine $|A \cap B|$.
(Q3) Given $A \in \{A, C, T, G\}^{m_1}, B \in \{A, C, T, G\}^{m_2}$ and $m_1 > m_2$, determine whether $B \subset A$
(Q4) Given a sequence $A \in \{A, C, T, G\}^m$ and a pattern $P \in \{A, C, T, G, ?\}^m$. Does A match pattern P?

The rest of article is organized as follows: In §II we provide some background and summarize the related work in the literature. Next, in §III we describe the protocols that we propose to answer the aforementioned questions with privacy. We evaluate our protocols and compare them with other state-of-the-art solutions in §IV. The article finishes in §V with a brief conclusion.

[1] http://www.genome.gov/10001772

[2] Very few alterations will appear throughout our lifetime

2372-9198/15 $3.00 © 2015 IEEE
DOI 10.1109/CBMS.2015.70

II. Related work

Techniques for data anonymization have been studied for a long time now. Statistical disclosure control has several effective solutions for the protection of personal data such as microaggregation [8], noise addition, rank swapping or suppression and generalisation and recent examples can be applied to data from a variety of fields from collaborative filtering data [9] to biomedical data [10], [11]. While it could be argued that traditional data anonymization techniques could be applied to DNA sequences, since they are highly correlated and, even if some parts are suppressed, they could be reconstructed from the others. In fact, when data anonymization techniques were applied to DNA sequences, it was shown that they were either inefficient [12], [13] or the accuracy and utility of the result were significantly reduced [14]. These shortcomings have fostered researchers to develope novel methods for processing and mining genome data [15].

In most of those techniques, apart from private string searching and comparison algorithms [16], [17], [18], [19], it can be found that the so-called *Private Set Intersection* (PSI) problem plays a central role. The PSI was problem introduced by Freedman, Nissim and Pinkas [20] and involves two parties that want to jointly calculate the intersection of their sets without revealing any additional information about their sets. The problem has received a lot of attention and many solutions have been proposed in the literature [21], [22], [23]. However, the work of De Cristofaro [24] can be considered a milestone in this field as it decreased the complexity to linear. Significant performance boost has been made from Huang et al. [25] using garbled circuits, from Dong et al. [26] using a specially tailored variation of Bloom filters, called garbled Bloom filters and Pinkas et al. [27] using Oblivious Transfer. Despite this performance boost, when these methods are applied to compute the intersection of DNA sequences, they are still inefficient due to the huge volume of data to be managed. With the aim to avert this problem, Dong et al. [6] and Kamara et al. [7] introduced server-assisted protocols that improve the performance. Unfortunately, even with this improvement, the cost is still substantial and the application of these protocols is not practical yet.

III. Our protocols

A. General Scheme and Main Actors

We consider a scenario with three entities: Alice, Bob and Trudy. Alice and Bob have their DNA sequences $\mathcal{S}_A$ and $\mathcal{S}_B$. They do not trust each other, but they would like to allow some queries over their data as long as they do not disclose their whole DNA sequences. To do so, they count on Trudy, a semi-trusted entity (i.e. honest but curious). Therefore, while they believe that Trudy will honestly perform the required operations on their data, and that she will not collude with none of them, due to her curiosity, they are both reluctant to hand over their raw data.

Without loss of generality, we assume that Alice is the initiator of the protocol and that the results will be returned to her. For the sake of clarity, we do not elaborate on the communication channels between Alice, Bob and Trudy. We assume that secure channels are established between them (*e.g.* HTTPS, SSH, or the like) therefore an external adversary could only intercept encrypted information that cannot be decrypted neither inadvertently altered.

We base our protocols on the encoding of DNA sequences using Bloom Filters (BF). It is well-known that BF might lead to false positives but the probability of that happening is well bounded and can be easily managed.

The BF created by Alice and Bob are scrambled before being send to Trudy, who will be able to perform the calculations while, at the same time, she cannot deduce any information about the result.

It is worth noting that if Alice and Bob decide to keep secret the parameters of their hash functions, Trudy would not be able to find their DNA sequences. Even under these constraints, Trudy could infer some information about them, *e.g.* the size of the intersection set. Hence, depending on the privacy preferences of Alice and Bob, the protocol could be improved even further in terms of performance[3].

B. Setting up the environment

Initially, Alice contacts Bob to generate a shared secret κ, by using e.g. Diffie-Hellman. Also, they agree upon the parameters of their Bloom filters, namely the length m, the k hash functions $h_1, h_2, \ldots, h_k$ for their sets, a parameter λ, a keyed pseudo-random permutation $\pi_x : \{0,1\}^{m+\lambda} \rightarrow \{0,1\}^{m+\lambda}$ and if $\lambda > 0$ a keyed pseudo-random function. The parameter λ is the obfuscation parameter.

In what follows, we will use a function $SeqBloom$, which takes as input the parameters to generate a Bloom filter: the length L of the filter, the hash functions and the probability of false positives; and a sequence $S = s_1 s_2 \ldots s_m$. First, we create an empty Bloom filter B of length L. Then, for $i \in \{1, 2, \ldots, m\}$ we compute the elements $t_i = s_i || i$, where "||" denotes concatenation and we append t_i to B. The output of $SeqBloom$ is the Bloom filter B. The role of $SeqBloom$ is to append all the elements of the sequence in the Bloom filter, while preserving their order. Note that our sequence contains only A, C, G and T, therefore, if we need to encode the sequence in a Bloom filter, we need to encode them in such a way that we know the symbol and its position.

C. The PSI protocol

Alice and Bob send Trudy an obfuscated version of their Bloom filters, and Trudy will return them the Bloom filter of their intersection. Note that if $BF(A \cap B), BF(A)$, and $BF(B)$ use the same m and hash functions, then with overwhelming probability $BF(A \cap B) = BF(A) \cap BF(B)$.

The PSI protocol goes as follows: Alice and Bob generate the Bloom filters of their sets, $Bloom_A$ and $Bloom_B$ respectively, using the aforementioned $SeqBloom$ procedure. If they

[3]The calculations that Trudy needs to perform are very lightweight and can be easily parallelized. When she computes the results, she can send them to Alice for decoding and recovery.

agree to obfuscate their results (i.e. $\lambda > 0$) then they both pad $Bloom_A$ and $Bloom_B$ with λ random bits to obtain $Bloom'_A$ and $Bloom'_B$.

Then, Alice and Bob send Trudy $M_A = \pi_\kappa(Bloom'_A)$ and $M_B = \pi_\kappa(Bloom'_B)$ respectively. Trudy may easily calculate $Bloom_I = M_A \cap M_B$ and send the result to both of them. All Alice and Bob have to do to recover the Bloom filter of their intersection is to compute $\pi_\kappa^{-1}(Bloom_I)$ and remove the last λ bits. Each of them may now query the received Bloom filter to recover their common elements.

D. Pattern matching protocol

The pattern matching protocol bears much resemblance with the aforementioned PSI protocol, as it uses the Bloom filters to store both the information that is going to be queried as well as the pattern to match.

Let as assume that Alice has a pattern $\mathcal{P}_A$ to match, $\mathcal{P}_A = p_1, p_2 \dots p_i, p_m \in \{A, C, G, T, ?\}$. For instance $\mathcal{P}_A = AC?C?$ means that if Bob's sequence has length 5 and it contains an "A" in the first position and a "C" in the second and fourth, then it matches the pattern. Note that we use the character "?" to denote a non-empty character that can take any of the values in $\{A, C, G, T\}$. Alice creates $Bloom_A$ using the $SeqBloom$ function to store all the characters which are not equal to the wildcard "?" character. Then Alice sends $M_A = \pi_\kappa(Bloom_A)$ to Trudy. Bob uses the same procedure as in the previous protocol and sends $M_B = \pi_\kappa(Bloom_B)$ to Trudy. Trudy checks whether $M_A \cap M_B = M_A$ and returns the result to Alice.

E. Sub-sequence matching protocol

Let us assume that Alice has a sub-sequence $\mathcal{P}_A = p_1, p_2, \dots, p_\ell$ of length ℓ, and she wants to check whether Bob's sequence contains it. Moreover we assume that she has agreed with Bob on a Pseudo Random Function (PRF) denoted by $\mathcal{F}(k, x)$, where k is the PRF key and x is the point at which the function is evaluated. Alice computes $M_A = \mathcal{F}(\kappa, \mathcal{P}_A)$ and sends it to Trudy. Given Bob's sequence $B = b_1, b_2, \dots, b_m$, he creates a set of messages $M_{B_i} = \mathcal{F}(\kappa, \{b_i \dots b_{i+\ell}\}, i \in \{1, m - \ell\}$ and sends them to Trudy. Trudy may now calculate : $\mathcal{S} = \{i : i \in \{1, m-\ell\}\ and\ M_A = M_{B_i}\}$ Depending on the agreed output, Trudy can return to Alice: a) $\mathcal{S}$, that is the indexes (i) where the sub-sequence is found; b) $|\mathcal{S}|$, that is the number of times the sub-sequence was found; or c) True (i.e. $|\mathcal{S}| > 0$) or False, depending on whether the sub-sequence was found or not.

IV. Evaluation of the protocols

With the aim to empirically prove the efficiency of our protocols, we have implemented them in Python 2.7 using the Murmur hash 3 to hash the values in the Bloom filters. The results we report next, were obtained over a single thread on a computer with an Intel Core i7-4790 CPU (3.60GHz) and 16GB of RAM, and running 64-bit Debian GNU/Linux 3.16.0-4. The times that are reported refer only to the execution time (the data exchange over the network is not considered). For the experiments, we generated random DNA sequences of lengths $10^6, 10^7$ and 10^8 and performed 100 tests for each length[4].

TABLE I
Comparison with improved methods of [27]. Runtime is given in seconds, and communication bandwidth in MB.

	Runtime	**Communication complexity**
DH-based FFC [28]	24.7	24.0
DH-based ECC [28]	44.1	3.8
RSA-based [24]	31.3	17.3
Yao [29], [25]	128.2	2,880
GMW [30]	161.9	3,200
Vector-MT GMW [27]	100.3	1,920
Garbled Bloom Filter [26]	16.1	216
Random Garbled Bloom Filter [27]	8.1	72.6
Set Inclusion [27]	3.3	13.4
Proposed	0.01156	0.336

TABLE II
Comparison with the server-aided protocol of [7]. Runtime is given in seconds and, communication bandwidth in MB.

Server-Aided Protocol of Kamara et al.				**Proposed**		**Pattern**	**Substring**
Set Size	Threads #	Comm.	Runtime	Comm	Runtime	**Matching**	**Search**
1M	20	10	1.652	5	0.6	0.15907	0.04031
10M	100	114	7	51	7	1.57624	0.35627
100M	100	1239	53	514	83	15.78026	4.01845

Tables I and II show a comparison of our protocols with the other state-of-the-art proposals. We have used the results reported by the authors of each method. In this sense, since the times are not obtained in the same computer/conditions, these results serve only to show that our proposal is many times faster and efficient, but not to take a precise comparison of the performance boost[5]. It can be observed that the results of our protocols clearly outperform the non-server-aided protocols that are provided in [27].

Comparing our results with the server-aided protocols of Kamara et al., it can be noticed that the communication cost has been halved. Note that in our PSI protocol each party sends $\frac{-n \log p}{\log^2 2} + \lambda$ bits instead of αn in [7], where α is the length of the ciphertext. More interestingly, the running times reported by Kamara et al. are comparable to our running times. However, Kamara et al. use multiple execution nodes, from 20 to 100, while our non-optimized implementation runs on a single thread. Hence, the performance boost of our protocols are clear.

Finally, we report the average times required to execute our private pattern matching and our private sub-sequence matching protocols for different lengths of DNA sequences. More precisely, in the private matching protocol, we created large patterns, so that they are at least 99% the same (the case of relatives). For the private sub-sequence matching case, we

[4]Clearly, one could extend this experiments for up to 10^9 elements, which is the actual size of a DNA sequence. However, this was not necessary since it is apparent that our protocol scales linearly with the number of elements.

[5]While the machine that we used in our experiments compared to the one used by Pinkas et al. in [27] has considerable higher specs, our implementation is far from optimized (e.g. use of scripting language, serial execution, etc.) . Nevertheless, one cannot clain that it is 300 times faster to compare the runtimes.

counted how many times a pattern appeared. Note that the length of the pattern plays a little role in our protocol, as it only removes a very small number of hashes. The results are obtained from a hundred searches and they clearly indicate the efficacy of our approach. Also, the linearity of our algorithms is clearly shown in the experimental results, all of which can be performed in the scale of seconds.

V. Conclusions

Many efforts have been devoted to the analysis of the human genome. As a result, DNA sequencing is nowadays possible at low costs. Also, we have the possibility to share our sequences and the research community could improve early diagnose and treatment of diseases. To do so, several data mining techniques to explore the human genome exist. However, guaranteeing individuals privacy during these mining processes is computationally costly and current state-of-the-art techniques do not provide practical solutions yet.

In this article we have proposed several protocols to obtain specific information about DNA sequences while guaranteeing the privacy of the individuals. Also, we have proven that our solution is very efficient, thus, opening the door to the private exploration of the human genome in a feasible and practical way. In our next steps, we will improve our implementation and the implement the other protocols, to provide a clear and unbiased comparison. Moreover, we plan to further explore the use of Bloom filters in the standard model, without the use of a trusted third party.

Acknowledgment

Agusti Solanas is partly funded by La Caixa Foundation through project "SIMPATIC: Intelligent, Autonomous and Private Monitoring System based on ICT" RECERCAIXA'12, and by the Government of Catalonia under grant 2014 SGR 537.

References

[1] A. D. Johnson and C. J. O'Donnell, "An open access database of genome-wide association results," *BMC medical genetics*, vol. 10, no. 1, p. 6, 2009.

[2] Y. Erlich and A. Narayanan, "Routes for breaching and protecting genetic privacy," *Nature Reviews Genetics*, vol. 15, no. 6, pp. 409–421, 2014.

[3] D. J. Benjamin, D. Cesarini, M. J. van der Loos, C. T. Dawes, P. D. Koellinger, P. K. Magnusson, C. F. Chabris, D. Conley, D. Laibson, M. Johannesson *et al.*, "The genetic architecture of economic and political preferences," *Proceedings of the National Academy of Sciences*, vol. 109, no. 21, pp. 8026–8031, 2012.

[4] E. Ayday, E. De Cristofaro, J. Hubaux, and G. Tsudik, "The chills and thrills of whole genome sequencing," *CoRR*, vol. abs/1306.1264, 2013.

[5] E. De Cristofaro, "Genomic privacy and the rise of a new research community," *Security & Privacy, IEEE*, vol. 12, no. 2, pp. 80–83, 2014.

[6] C. Dong, L. Chen, J. Camenisch, and G. Russello, "Fair private set intersection with a semi-trusted arbiter," in *Data and Applications Security and Privacy XXVII*. Springer, 2013, pp. 128–144.

[7] S. Kamara, P. Mohassel, M. Raykova, and S. Sadeghian, "Scaling private set intersection to billion-element sets," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, N. Christin and R. Safavi-Naini, Eds. Springer Berlin Heidelberg, 2014, pp. 195–215. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-45472-5_13

[8] A. Solanas and A. Martínez-Ballesté, "V-mdav: Variable group size multivariate microaggregation," *COMPSTAT 2006*, pp. 917–925, 2006.

[9] F. Casino, J. Domingo-Ferrer, C. Patsakis, D. Puig, and A. Solanas, "A k-anonymous approach to privacy preserving collaborative filtering," *Journal of Computer and System Sciences*, 2014.

[10] A. Solanas, A. Martínez-Ballesté, and J. M. Mateo-Sanz, "Distributed architecture with double-phase microaggregation for the private sharing of biomedical data in mobile health," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 6, pp. 901–910, 2013.

[11] A. Zigomitros, A. Solanas, and C. Patsakis, "The role of inference in the anonymization of medical records," in *2014 IEEE 27th International Symposium on Computer-Based Medical Systems, New York, NY, USA, May 27-29, 2014*, 2014, pp. 88–93.

[12] S. Sankararaman, G. Obozinski, M. I. Jordan, and E. Halperin, "Genomic privacy and limits of individual detection in a pool," *Nature genetics*, vol. 41, no. 9, pp. 965–967, 2009.

[13] M. Gymrek, A. L. McGuire, D. Golan, E. Halperin, and Y. Erlich, "Identifying personal genomes by surname inference," *Science*, vol. 339, no. 6117, pp. 321–324, 2013.

[14] M. Humbert, E. Ayday, J.-P. Hubaux, and A. Telenti, "Addressing the concerns of the lacks family: Quantification of kin genomic privacy," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 1141–1152.

[15] G. Loukides, A. Gkoulalas-Divanis, and B. Malin, "Anonymization of electronic medical records for validating genome-wide association studies," *Proceedings of the National Academy of Sciences*, vol. 107, no. 17, pp. 7898–7903, 2010.

[16] M. Blanton and M. Aliasgari, "Secure outsourcing of dna searching via finite automata," in *Data and Applications Security and Privacy XXIV*. Springer, 2010, pp. 49–64.

[17] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik, "Countering gattaca: efficient and secure testing of fully-sequenced human genomes," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 691–702.

[18] C. Blundo, E. De Cristofaro, and P. Gasti, "Espresso: efficient privacy-preserving evaluation of sample set similarity," *Journal of Computer Security*, vol. 22, no. 3, pp. 355–381, 2014.

[19] M. Kantarcioglu, W. Jiang, Y. Liu, and B. Malin, "A cryptographic approach to securely share and query genomic sequences," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 12, no. 5, pp. 606–617, 2008.

[20] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Advances in Cryptology-EUROCRYPT 2004*. Springer, 2004, pp. 1–19.

[21] A. Kiayias and A. Mitrofanova, "Testing disjointness of private datasets," in *Financial Cryptography and Data Security*. Springer, 2005, pp. 109–124.

[22] C. Hazay and Y. Lindell, "Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries," in *Theory of Cryptography*. Springer, 2008, pp. 155–175.

[23] J. Camenisch and G. M. Zaverucha, "Private intersection of certified sets," in *Financial Cryptography and Data Security*. Springer, 2009, pp. 108–127.

[24] E. D. Cristofaro and G. Tsudik, "Practical private set intersection protocols with linear complexity," in *Financial Cryptography*, 2010, pp. 143–159.

[25] Y. Huang, D. Evans, and J. Katz, "Private set intersection: Are garbled circuits better than custom protocols," in *Network and Distributed System Security Symposium (NDSS). The Internet Society*, 2012.

[26] C. Dong, L. Chen, and Z. Wen, "When private set intersection meets big data: An efficient and scalable protocol," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 789–800.

[27] B. Pinkas, T. Schneider, and M. Zohner, "Faster private set intersection based on ot extension," *USENIX Security*, 2014.

[28] B. A. Huberman, M. Franklin, and T. Hogg, "Enhancing privacy and trust in electronic communities," in *Proceedings of the 1st ACM conference on Electronic commerce*. ACM, 1999, pp. 78–86.

[29] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, "Efficient garbling from a fixed-key blockcipher," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 478–492.

[30] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More efficient oblivious transfer and extensions for faster secure computation," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 535–548.