Privacy Threats and Practical Solutions for Genetic Risk Tests

Ludovic Barman, Mohammed-Taha Elgraini, Jean Louis Raisaro, and Jean-Pierre Hubaux
École Polytechnique Fédérale de Lausanne
Lausanne, Switzerland
{*firstname.lastname*}*@epfl.ch*

Erman Ayday
Bilkent University,
Ankara, Turkey
erman@cs.bilkent.edu.tr

***Abstract*—Recently, several solutions have been proposed to address the complex challenge of protecting individuals' genetic data during personalized medicine tests. In this short paper, we analyze different privacy threats and propose simple countermeasures for the generic architecture mainly used in the literature. In particular, we present and evaluate a new practical solution against a critical attack of a malicious medical center trying to actively infer raw genetic information of patients.**

***Keywords*-genomic privacy; risk tests; cryptography;**

I. Introduction

Recent developments in genome sequencing, in particular the drastic reduction in sequencing costs, have enabled an increasing use of genomic data for healthcare. At the same time, as discussed in many studies [1], [2], genomic information is highly sensitive because it reveals information about an individual's ethnicity, kinship, predisposition to diseases, phenotype, etc. Therefore, there is a strong need to protect the privacy of individuals' genomic data.

One way to achieve this protection is via encryption and there have been several systems proposed to securely store and allow for operations on encrypted genomic data [3], [4], [5], [6]. In this work, we focus on the systems that provide genetic risk tests in a privacy-preserving way. One common property of these systems is their architecture and the parties involved. More specifically, such systems generally assume the existence of (i) a trusted institution, responsible for the sequencing, (ii) a data center, responsible for storing the genomic data and performing some operations on it, (iii) one or more medical centers (e.g., a physician, a pharmaceutical company) that prescribe and perform the genetic tests, and (iv) the individual (or the patient) who provides his genomic information upon consent.

In this paper, sticking to this generic architecture, we first analyze the privacy issues of the existing solutions and demonstrate tractable attacks. Then, we propose efficient and practical countermeasures. In particular, we propose a new protocol against an attack carried out by a potentially malicious medical center trying to learn the genetic data of the patient (rather than the result of the test) by forging some of the test parameters. Our protocol allows the data center (which is unaware of the nature of the genetic test) to iteratively check some parameters of the test until it is convinced the ongoing test is not an attack. The protocol introduces a tradeoff between privacy and practicality. In other words, it leaks minimum information about the test to the data center to provide a practical protocol.

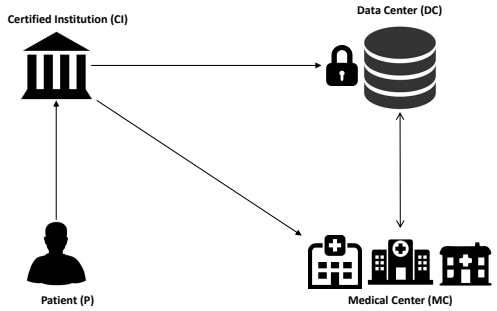

Figure 1: System model architecture. P's genome is sequenced by the CI, stored encrypted at the DC, and the MCs are able to perform risk tests on it. The CI is responsible for distributing the keys, and the MC can get only the final result of the test computation.

The rest of the paper is organized as follows. In Section II, we give an overview of the system model and genetic disease risk test between the involved parties. In Section III, we provide a taxonomy of different privacy threats, as well as some basic countermeasures. In Section IV, we propose an efficient protocol against a potentially malicious medical unit. In Section V, we study the tradeoff between the practicality and the privacy of the proposed protocol. Finally, in Section VI we conclude the paper.

II. System Model

In this section, we describe the generic architecture (Fig. 1), usually found in the literature, for performing genetic risk tests in a privacy-preserving way. In short, the genetic information of the *patient* (P) is processed at a trusted *certified institution* (CI), which sequences the DNA, extracts the SNPs, encrypts them, and distributes the cryptographic keys to the other parties. A *data center* (DC) is used to store the encrypted genetic information, while one or several *medical center(s)* (MC) can compute genetic risk tests for P in a privacy-preserving way. In clinical care, the genetic risk G is usually computed as a weighted sum of the raw SNPs' values, $G = \sum(\beta_i \times SNP_i)$, where β_i is the contribution (or SNP weight) of SNP_i to the risk. Such a computation is performed, in a privacy-preserving

way, via secure multiparty computation or using a trusted hardware (such as a smart card). The MC learns only the final result, but not the SNPs' raw values. In addition, the DC does not learn the SNP weights used in the test because the computation is performed at the MC.

III. Threats and Countermeasures

In this section, we investigate the different privacy issues and propose potential countermeasures for the architecture presented in Section II. Previous works consider the P and the CI to be trusted parties, and they assume the DC and the MC(s) to be honest-but-curious parties. Such an assumption can be considered too strong for a real-life scenario. Hence, in our analysis, we extend this threat model beyond the honest-but-curious case, by focusing on other possible behaviors for both the DC and the MC.

We assume that a party could be either (i) *honest* when it perfectly abides by the protocol, (ii) *semi-honest* (or *honest-but-curious*, or *passive*) when it honestly follows the protocol specification without altering the data but still tries to infer sensitive information that should remain private, or (iii) *dishonest* (or *malicious*, or *active*) when it can arbitrarily deviate from the protocol specification. In the last category, we will focus on the interesting case of a *dishonest-but-covert* adversary that is willing to actively cheat, but only if he is not caught [7].

As shown in Fig. 2, we can distinguish three main attacks depending on the threat model we consider. Note that more sophisticated attacks can also be performed in this scenario. We decided to focus on the most likely and simple ones.

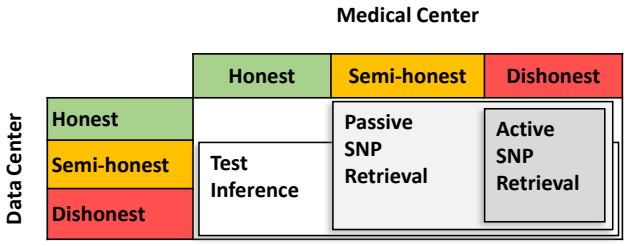

Figure 2: Table showing the different threat models and the three main attacks that can be performed.

A. Test Inference Attack

A *test inference* attack can be performed by a semi-honest data center that might try to infer the nature of the ongoing test for a given patient *P*. By looking at the MC's test request, the DC learns which SNPs are accessed by the MC, how many SNPs are used, and how often. For instance, the DC can infer that *P* is suffering from heart disease, if the ongoing test involves SNPs correlated with cardiovascular risk. Such an attack can dramatically jeopardize P's privacy if the DC also has access to some contextual information that can re-identify P. Re-identification of individuals has been extensively studied in the literature [8], [9], [10], and it has been proved to be even more effective when the attacker has either some additional genetic information about the target (different from the one securely stored at the DC) or about his relatives, or some background knowledge about the test request (i.e., location, and time).

Possible Countermeasure(s): A simple and naïve approach to address this attack is to use the entire database for each genomic risk test. In this way, the DC cannot infer the number and the type of SNPs used for the test. Following this idea, in [5], Danezis *et al.* propose a method that computes the genetic risk test for a given patient by involving all his SNPs in the computation. The SNPs that do not contribute to the genetic risk are canceled out from the final result by using *zero* weights. However, it is easy to argue that such a solution does not scale for large databases, especially if we consider the increasing discovery rate of new SNPs with clinical relevance. A more sophisticated approach for hiding MC's access patterns is represented by the use of one of the following techniques: Oblivious RAM (ORAM) [11], or private information retrieval (PIR) [12]. In [4], to protect MC's access patterns from DC, Karvelas *et al.* propose the use of ORAM. Yet, the main drawback of such a solution is that the protocol periodically requires the entire database to be reshuffled, which causes significant computational overhead. In addition, ORAM allows the MC to obliviously write data on the DC, which is an overkill in this scenario, and could represent a significant overhead. Similar concerns also characterize PIR. Even though both techniques have been proved to be more efficient than the naïve solution, they are still very expensive, unpractical and hard to implement in real-life scenarios. Therefore, we propose in Section IV a more practical solution that enables the MC to calibrate its strategy according to its privacy/efficiency preferences.

B. Passive SNP Retrieval Attack

A *passive SNP retrieval* attack can be performed by a semi-honest MC that tries to infer P's SNPs' raw values from the test end-result. Note this attack and the next one are meaningful if we assume that the MC is not allowed to ask the DC for the raw values of the SNPs but can obtain only the end-result of a test. As the MC knows the characteristics of the exposed SNPs (e.g., linkage disequilibrium (LD) between SNPs), and the SNP weights, the risk computation can be seen as a linear equation where the SNPs' values are the unknowns. P's privacy decreases with the number of tests performed by the MC that can build an overdetermined system of linear equations and easily solve it.

Possible Countermeasure(s): Note that there is no perfect solution for preventing such a brute-force attack. Yet, a potential mitigation has been proposed in [3] where the final test result is provided as a range. Obviously, as the range size increases, the utility at the MC decreases, but P's genomic privacy decreases slower. The optimal range size for a test

result might change based on the test to be performed. It is necessary to strike a balance between clinical utility and patients' privacy.

C. Active SNP Retrieval Attack

An *active SNP retrieval* attack can be performed by a dishonest MC that arbitrarily deviates from the protocol and sets up new SNP weights for a given test in such a way as to easily retrieve, from the test end-result, SNPs' raw values and compromise patients' genomic privacy. Let G be the end-result of a genetic risk test computed as described in Section II. By setting $\beta_i = 0\ \forall i$ except for one $\beta_j = 1$, where $i \neq j$, the MC gets that G is trivially equal to SNP_j. Note that such an attack can be reiterated until the MC has obtained all the raw SNPs of the target patient. In a smarter version of the same attack, a malicious MC can create weights as consecutive powers of a number, ρ, greater than the highest value used for encoding the SNPs. In our case, we assume that the *additive model* has been used to encode the SNPs. According to such an encoding, a SNP value corresponds to 0 if it is homozygous major, to 1 if it is heterozygous, or to 2 if it is homozygous minor. Hence, ρ should be greater than 2. Now, assume that the attacker sets the SNP weights as a ternary base such that $\beta_i = 3^i$. Consider a test of size $N = 3$ and the following weights: $\beta_0 = 3^0 = 1$, $\beta_1 = 3^1 = 3$, and $\beta_2 = 3^2 = 9$. If the final results in base 10 of the risk test computation yields to $G_{b10} = 22$, its equivalent in base 3 is $G_{b3} = 211$. As $G = SNP_2 \times \beta_2 + SNP_1 \times \beta_1 + SNP_0 \times \beta_0$, the attacker can easily map each SNP to the corresponding digit and immediately obtain the raw values: $SNP_2 = 2$, $SNP_1 = 1$, and $SNP_0 = 1$.

Possible Countermeasure(s): We emphasize that such an attack could represent the most likely and dangerous threat for the genomic privacy of patients in a real-life scenario, mainly for two reasons: (i) It is easy to perform in practice, and (ii) it is hard to check whether the protocol is followed honestly by the multitude of the MCs that could potentially make use of the system. Indeed, given the design of the system where the SNP weights are kept private at the MC, the DC has no way to check if a malicious MC is trying to run a legitimate test or an attack. In other words, if it tries to actively retrieve the SNPs by arbitrarily setting the SNP weights, a dishonest MC is fully covert and it gets caught with null probability. A possible mitigation to this attack is to make a compromise between the privacy of the MC (i.e., the SNP weights used for a given test) and the privacy of patients' genetic information stored at the DC. As such, we propose in Section IV a protocol where the MC needs to convince the DC, before getting the final result of the genetic risk test, that its weights are legitimate and not meant to reveal raw SNPs.

IV. Proposed Solution

We saw in Section III that if the MC can freely set the SNP weights for a given test, it can efficiently recover the SNPs' raw values, without being detected. We propose a practical solution for the *active SNP retrieval* attack, where the MC is forced to play fair by iteratively revealing some of the SNP weights to the DC, until the DC is convinced that the ongoing test is legitimate. The proposed solution slightly compromises the MC's privacy and gives more capabilities to the (potentially malicious) DC. Indeed, the test parameters might allow the DC to infer the nature of the test (as seen in Section III-A), and also might represent some valuable private information in the eyes of the MC.[1] Note that the MC can abort the protocol at any time, if it considers the information leakage on the test parameters being above a given threshold. As previously done in other works, [3], we assume that the encrypted SNPs are stored shuffled at the DC and only the MC knows the mapping that is maintained by the CI. The protocol is based on the scheme discussed in Section II, and can be described as follows:

1) The MC wants to compute a genetic risk tests on a given patient involving R SNPs and to convince the DC that his SNP weights are legitimate. To confuse the DC, and to prevent him from a *test inference* attack based on the number of SNPs requested, it pads his request with D supplementary dummy SNPs. Dummy SNPs are not related to the test and their contribution is cancelled out from the test computation by *zero* weights.[2] Let $N = R + D$ be the total length of the request including real and dummy SNPs.
2) The MC sends to the DC the request for N SNPs, and a commitment for each SNP weight, β_i, that will be used in the test computation (both *zero* weights and *non-zero* weights). Let $C_i = Commit(\beta_i)\ \forall i \in [0, N-1]$ be such a commitment.
3) The DC randomly picks two indices j, $k \in [0, N-1]$, and sends them to the MC. As a response, the MC sends the correspondent β_j and β_k, in clear, to the DC.
4) The DC verifies the commitments C_k and C_j, and then checks the value of the two SNP weights. If both β_j and β_k are non-zero, and not different powers of the same number, the DC is convinced that the ongoing test is not an *active SNP retrieval* attack. Steps 3 and 4 are repeated until the DC is convinced, or the MC aborts the protocol. At each new iteration, except the first, the DC asks the MC for only one new weight.

[1] It is possible, for example, that a MC, being represented by a pharmaceutical company, does not want to reveal the coefficients used for a pharmacogenetic test on a drug under development, before the test is patented.

[2] Padding the request with the entire set of SNPs stored at the DC reduces to the naïve solution described in Section III-A.

5) Once convinced, the DC sends to the MC the S encrypted SNPs corresponding to the weights not seen during the previous steps. These SNPs are at most $S = N - 2$, if it is convinced at the first iteration.
6) Using the S encrypted SNPs, the MC homomorphically computes the encryption of the first part of the test result, $\text{ENC}(G_1)$, and sends it to the DC.
7) The DC computes the encryption of the second part of the test result $\text{ENC}(G_2)$, based on the other encrypted SNPs for which it knew the weights at steps 3 and 4. The two partial results are homomorphically added, such that $\text{ENC}(G) = \text{ENC}(G_1) + \text{ENC}(G_2)$. The final encrypted result, ENC(G), is partially decrypted to $\text{ENC}(\hat{G})$ and sent to the MC.
8) The MC performs the final decryption of $\text{ENC}(\hat{G})$ to obtain the final test result, G. The protocol ends.

Note that once the DC is convinced from the SNP weights revealed by the MC, it computes the encryption of the partial test result, $\text{ENC}(G_2)$ to make sure that, whatever other weight the MC decides to use for $\text{ENC}(G_1)$, an *active SNP retrieval* attack cannot be performed. Also note that the proposed solution protects against this specific attack. More sophisticated attacks involving multiple requests of the same SNPs on a given patient, or collusion of multiple MCs can only be prevented by enforcing an access control infrastructure maintained by the CI. We will study such a countermeasure in future work.

V. Evaluation and Discussion

As discussed in Section IV, for the protocol to be successful and to preserve patients' genomic privacy against an *active SNP retrieval* attack, the MC leaks some information about the test. We quantified the information leakage at the MC, depending on the request length (N), i.e., the number of real SNPs (i.e., the test length R) along with the number of dummy SNPs (D). As shown in Fig. 3, the colored curves, drawn from an hypergeometric distribution, represent the expected rate of real SNP weights leaked from the MC, given the number of iterations performed to convince the DC about the legitimacy of the test. The diamonds show when DC is convinced. We can observe that the leakage of real SNP weights is linear and slower with a higher number of dummy SNPs. In addition, independently from the number of dummies used by the MC in the request, the leakage is always around 2 real SNP weights. Note that if we want to minimize the request latency at the MC, the use of dummies seems useless.

However, the leakage of real SNPs weights is not the only important metric; the test length is also an important information for an attacker at the DC trying to infer the nature of the ongoing test. As such, we used the Shannon entropy to measure the uncertainty on the test length of the attacker or, in other words, the privacy level of the MC. We implemented the *convincing* protocol (steps 3 and 4 of the proposed solution) in Matlab and, as the DC checks the weights randomly, we simulated it 500 times to smooth out the results.

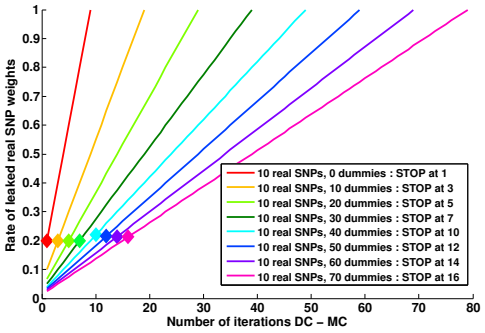

Figure 3: Evolution of the rate of real SNP weights leaked, given the number of iteration done between the MC and the DC for the *convincing* protocol.

As expected, we observe from Fig. 4, that the more dummy SNPs are requested from the MC, the slower the curves decay towards zero, meaning that the uncertainty of the adversary increases as it increases the number of dummies.

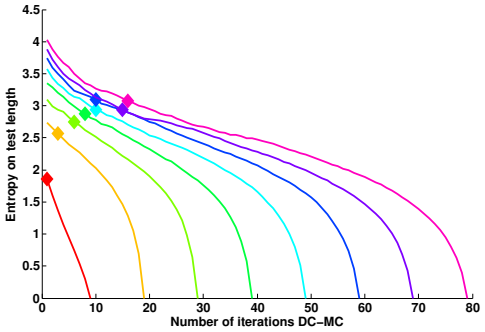

Figure 4: Evolution of the loss of entropy of test size of real SNP weights leaked, given the number of iteration done between the MC and the DC for the *convincing* protocol.

The tension between privacy and efficiency is obvious. The best tradeoff on the number of dummy SNPs to use in the request depends on the context of the MC. For instance, on one hand, if we consider the MC to be a pharmaceutical company trying to make pharmacogenetic risk tests for a clinical trial on a population of thousands of patients, the best strategy will be not to add dummies, as it will slow down the process and make the computation more intensive. On the other hand, if we consider the example of a critical genetic risk test run on a well-known personality, the MC will not want to take any risk in leaking the nature of the test from its length. Hence, the best strategy is to maximize the number of dummy SNPs in the request.

VI. Conclusion

In this paper, we have analyzed the privacy threats of a generic model used for privacy-preserving genetic risk tests. We have focused on the case of a dishonest-but-covert

MC wanting to actively infer important information about patients by purposefully modifying the test SNP weights. We have proposed a practical solution that prevents such an attack by checking the fairness of the MC. We emphasize that we have focused on a practical and usable solution for real-life scenarios, rather than designing a perfect but costly scheme. In the same line of thought, further improvements can be made in future work. In particular, access control is a must for further enhancing the protection of sensitive genetic information. Hiding the access patterns, possibly through an efficient PIR, would be an additional improvement over the obfuscation done when accessing data at the DC.

REFERENCES

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

[2] E. Ayday, E. D. Cristofaro, G. Tsudik, and J. P. Hubaux, "The chills and thrills of whole genome sequencing," *IEEE Computer Magazine*, 2013.

[3] E. Ayday, J. L. Raisaro, J. Rougemont, and J.-P. Hubaux, "Protecting and evaluating genomic privacy in medical tests and personalized medicine," in *Proceedings of the 12th Workshop on Privacy in the Electronic Society*. ACM, 2013.

[4] N. P. Karvelas, A. Peter, S. Katzenbeisser, E. Tews, and K. Hamacher, "Privacy preserving whole genome sequence processing through proxy-aided oram," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. ACM, 2014.

[5] G. Danezis and E. De Cristofaro, "Fast and private genomic testing for disease susceptibility," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. ACM, 2014, pp. 31–34.

[6] M. Djatmiko, A. Friedman, R. Boreli, F. Lawrence, B. Thorne, and S. Hardy, "Secure evaluation protocol for personalized medicine," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. ACM, 2014, pp. 159–162.

[7] Y. Aumann and Y. Lindell, "Security against covert adversaries: Efficient protocols for realistic adversaries," in *Theory of Cryptography*. Springer, 2007, pp. 137–156.

[8] 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.

[9] M. Humbert, E. Ayday, J.-P. Hubaux, and A. Telenti, "Addressing the concerns of the lacks family: Quantification of kin genomic privacy," in *CCS*, 2013.

[10] B. Malin and L. Sweeney, "How (not) to protect genomic data privacy in a distributed network: Using trail re-identification to evaluate and design anonymity protection systems," *Journal of Biomedical Informatics*, vol. 37, no. 3, pp. 179–192, 2004.

[11] O. Goldreich, "Towards a theory of software protection and simulation by oblivious rams," in *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. ACM, 1987, pp. 182–194.

[12] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *Journal of the ACM (JACM)*, vol. 45, no. 6, pp. 965–981, 1998.