Fast and Private Genomic Testing for Disease Susceptibility

George Danezis
Department of Computer Science
University College London

Emiliano De Cristofaro
Department of Computer Science
University College London

ABSTRACT

Advances in DNA sequencing are bringing mass computational genomic testing increasingly closer to reality. The sensitivity of genetic data, however, prompts the need for carefully protecting patients' privacy. Also, it is crucial to conceal the test's specifics, which often constitute a pharmaceutical company's trade secret. This paper presents two cryptographic protocols for privately assessing a patient's genetic susceptibility to a disease, computing a weighted average of patient's genetic markers (the "SNPs") and their importance factor. We build on the architecture introduced by Ayday et al. but point out an important limitation of their model, namely, that the protocol leaks which and how many SNPs are tested. Then, we demonstrate that an alternative SNP encoding can simplify (private) computations, and make patient-side computation on a smartcard device extremely efficient. A second protocol variant, based on secret sharing, further reduces online computation.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Secure Computation

Keywords

Privacy, Genomics, Cryptographic Protocols

1. INTRODUCTION

In the past few years, progress in DNA sequencing genomics has been quite exceptional, with costs dropping faster than what Moore's law would predict. This enables important advances not only in genetics, but also in healthcare, as genome-based medicine becomes increasingly preventive and personalized [10]. At the same time, however, genomic data disclosure prompts important privacy and ethical concerns. Genomes not only uniquely and irrevocably identifies their owner, but also contain information about ethnic heritage and susceptibility to diseases and conditions (including mental disorders), thus raising fears of genetic discrimination [1]. Due to its hereditary nature, disclosing one's genome implies disclosing that of close relatives too [11] and masking sensitive portions of the genome is essentially impossible as correlation between genetic mutations (aka linkage disequilibrium) can be used to reconstruct redacted features [9]. Aiming to address these concerns, the research community has designed and prototyped a few cryptographic techniques realizing privacy-preserving in-silico genomic testing, such as [2–4, 6–8, 12, 14].

WPES'14, November 3, 2014, Scottsdale, Arizona, USA.

http://dx.doi.org/10.1145/2665943.2665952.

Private Disease Susceptibility (PDS). We focus on privately assessing a patient's susceptibility to a given disease, i.e., without revealing anything other than the test outcome. Susceptibility is computed as a weighted average, depending on whether some genetic markers – aka Single Nucleotide Polymorphisms (SNPs) – are expressed in patient's genome, and some importance factors. Specifically, susceptibility S to disease X is computed as:

$$S(X) = \frac{\sum_i C_i \cdot \Pr[X|\text{SNP}_i]}{\sum_i C_i},$$

where, for each of SNP_i, C_i is the importance factor and $\Pr[X|\text{SNP}_i \in \{0, 1, 2\}]$ a SNP-dependent weight. 0, 1, 2 denote, respectively, the presence of the SNP in no, one, or both chromosomes.

In [3], Ayday, Raisaro, Hubaux, and Rougemont presented a cryptographic protocol, denoted as **ARHR13** in the rest of the paper, relying on the following model: A *Certified Institution (CI)* receives a *Patient (P)*'s genetic sample, sequences it, and produces an encrypted encoding of all possible SNPs. The encrypted sequence is then stored on another (cloud-based) entity called *Storage and Processing Unit (SPU)*. Patients are issued with a smartcard, that contains part of the secret key needed to decrypt the genetic information. When a test is to be performed by a *Medical Centre (MC)*, the MC initiates a protocol involving the SPU (holding the encrypted data), P's smartcard (holding a secret key), and the MC itself (which holds the weights). In other words, the ARHR13 protocol is so that the results of the test are calculated over encrypted data to protect P's privacy. Additionally, weights and importance factors are not revealed to the patient for trade secrecy.

Issues with ARHR13. One limitation incurred by ARHR13 is the use of the expensive Bresson, Catalano, and Pointcheval (BCP) [5] variant of the Paillier cryposystem. Another, more fundamental issue is that the protocol actually reveals *which* and *how many* SNPs are being tested. As pointed out in [8], this may raise a few issues: (1) an eavesdropper learns how many SNPs are being tested and can infer which disease or condition the patient is being tested for; (2) the SPU learns the exact SNPs used and can therefore infer which condition is being tested; (3) the identity and size of the SNPs involved in a test, i.e., not only weights and importance factors, might also constitute the pharmaceutical company's trade secret, and thus should not be disclosed. To obviate this issue, one could use padding, i.e., adding some fictitious SNPs, with zero weight, to the test. However, this padding should then amount to all possible "interesting" SNPs, which is in the order of 1 million. Unfortunately, ARHR13 does not scale well enough to support this extension, due to the un-

derlying BCP cryptosystem's inefficiency. As we report in Sec. 5, running times are in the order of 1 hour on commodity hardware, and gigabytes of (encrypted) traffic would need to be exchanged.

Roadmap. Motivated by the above issues, this short paper revisits the Private Disease Susceptibility (PDS) protocol proposed by Ayday, Raisaro, Hubaux, and Rougemont (ARHR13) [3]. We propose a much simpler, yet equivalent, encoding for the SNPs, which enables us to design two significantly more efficient protocol variants: the first relies on the user-side smartcard to process (part of) the computation, and the second on infrastructure servers. Our novel protocols fulfill all requirements of the original proposal, and their efficiency allows us to extend the scheme to hide *which* and *how many* SNPs are being tested. As we show in Sec. 5, our protocols run in the order of minutes even when testing 1 million SNPs, and only generate a few megabytes of traffic.

2. PRELIMINARIES

2.1 The ARHR13 Protocol

We now review the Private Disease Susceptibility (PDS) protocol proposed in [3] (denoted as ARHR13 in the rest of the paper):

• **Step 0:** Cryptographic keys (public and secret keys) of each patient are generated and distributed to the patients. Symmetric keys are also established between all parties.

• **Step 1: (P → CI)** The patient (P) provides a sample to the Certified Institution (CI) for sequencing.

• **Step 2: (CI → SPU)** The CI performs sequencing and encrypts patient's real SNP using a symmetric key k_{PC} shared with P. (If at least one allele carries the mutation, the SNP is "*real*" and "*potential*" otherwise). It also encrypts patient's real and potential SNP positions under P's public key, using a modified Paillier cryptosystem [5] (which, besides additive homomorphism can also support proxy re-encryption). P's public key is denoted as $(n, g, h = g^x)$, where the strong secret key is the factorization of $n = pq$, the weak secret key is $x \in [1, n^2/2]$. The encryption of a real SNP_i, under P's public key, results in $E(\text{SNP}_i, g^x)$ and $E(\text{SNP}_i^2, g^x)$. (The second ciphertext is needed to carry out homomorphic operations later on.) Also, the CI encrypts an arbitrary position v_0 for every potential SNP, using k_{PC}, and sends the encrypted SNPs of P to the SPU.

• **Step 3: (P → MC, SPU)** The patient's weak secret key x is divided into two shares: $x^{(1)}$ and $x^{(2)}$ (such that $x = x^{(1)} + x^{(2)}$). Then, $x^{(1)}$ is given to the SPU and $x^{(2)}$ to the MC in the next step. Note that, thanks to the proxy re-encryption property, an message encrypted under P's public key can be partially decrypted by the SPU using $x^{(1)}$, and then decrypted at the MC using $x^{(2)}$ to recover the original message.

• **Step 4: (MC → P)** The MC wants to conduct a susceptibility test on P to disease X. P provides the other part of his secret key $x^{(2)}$ to the MC, which tells the patient the positions of the SNPs required for the test and obtains his consent to run the test.

• **Step 5: (P → SPU)** The patient's smartcard encrypts each requested position using the symmetric key shared with the CI. The patient sends the SPU the encrypted positions of the requested SNPs.

• **Step 6: (SPU → MC)** The SPU receives each requested position in an encrypted form. If the patient has a real SNP at the requested position, the SPU retrieves the encrypted SNP at the corresponding (encrypted) location. Otherwise, the SPU retrieves an encryption of v_0. Then, the retrieved SNPs are re-encrypted at the SPU under the P's public key. An encrypted SNP (using a random $r \in [1, n/4]$) is re-encrypted, under the same public key, by using a new random number $r_1 \in [1, n/4]$. Re-encrypted (or partially decrypted) SNPs are sent to MC (by SPU) in the same order as they are requested.

• **Step 7: (MC → SPU)** The MC computes P's total susceptibility for disease X by using the homomorphic properties of the cryptosystem. The MC sends the encrypted result to the SPU.

• **Step 8: (SPU → MC)** The SPU partially decrypts the result using $x^{(1)}$ with proxy re-encryption and sends it back to the MC.

• **Step 9:** The MC decrypts the message received from the SPU by using $x^{(2)}$ and recovers the result.

2.2 Simpler Encoding of the SNPs

Our first observation is that each SNP_i can only be one of 3 distinct values, i.e., 0, 1 or 2. The original ARHR13 scheme encodes those integers directly. On the contrary, we propose to encode each SNP as a 3-bit binary vector, with a value of 1 corresponding to the observed SNP, and the value of zero at other positions (for example SNP_i = 0, 1 or 2 would be represented as 100, 010 or 001, respectively). We then note that, for each binary indicator variable $I_j \in \{0, 1\}$ representing a possible value at a certain $\text{SNP}_i = \alpha$, we can associate a weight $w_j = C_i \cdot \Pr[X|\text{SNP}_i = \alpha]$. For a test constant $Z = \sum_i C_i$, the computation reduces to:

$$S = \frac{\sum_j w_j I_j}{Z}$$

Intuition. The encoding of SNPs as binary indicator variables removes the need for non-linear operations such as squarings that were used in the ARHR13 protocol. The computation is reduced to a simple sum of pairs of secret values – one provided by the MC (or pharmaceutical company) representing the test (w_j) and one provided by the SPU and representing the genome of the user (I_j). We note that the encodings of the SNP as $\text{SNP}_i \in \{0, 1, 2\}$ or as indicator variables $I_j \in \{0, 1\}$ contain exactly the same information – and given the one in clear, the other may be produced.

The new encoding allows us to replace the expensive BCP cryptosystem [5] with a significantly faster Additively Homomorphic Elliptic Curve based El-Gamal (AH-ECC) cryptosystem (presented below). Also note that ciphertexts produced by AH-ECC are significantly shorter than BCP encryptions – see Sec. 5.

2.3 Cryptography Background

We rely on the Additively Homomorphic Elliptic Curve based El-Gamal (AH-ECC) cryptosystem, which involves three algorithms:

1. *KeyGen*(1^τ): On input a security parameter τ, select an appropriate elliptic curve E and (G, H) public generators on E (generating a group of order q). Choose a random private key $x \in Z_q$, define the public key as $pk = x \cdot G$, and output public parameters (E, G, H, pk) and private key x.
2. *Encrypt*(m, pk): Message m is encrypted by drawing a random element $k \in Z_q$ and computing two EC-points as $(A, B) := (k \cdot G, k \cdot (pk) + m \cdot G)$. The output ciphertext is (A, B).
3. *Decrypt*(A, B, x): Decryption is performed by computing the element $m \cdot G = B - x \cdot A$. A pre-computed table of discrete logarithms may then be used to recover m from $m \cdot G$ (which is practical for small ranges of m).

Pairwise point addition of ciphertexts yields an encryption of their sum, i.e., $v \cdot E[a] + w \cdot E[b] = E[v \cdot a + w \cdot b]$. Ciphertexts may also be re-randomized by adding a fresh ciphertext of zero.

3. PRIVATE DISEASE SUSCEPTIBILITY TESTING BASED ON SMARTCARDS

We now show how the computation on the SNPs may be facilitated, compared to ARHR13 [3], by some user-held trusted hardware, such as a smartcard, using the encoding outlined in Sec. 2.2.

Assumptions. Similar to ARHR13 [3], we assume that the patient's SNPs are encrypted, upon sequencing, by the Certified Institution (CI) using standard symmetric encryption (e.g., AES in CBC mode, with a MAC using an encrypt-then-MAC mode), under a key K. Genomic data is also to be signed by the CI to certify its origin. Encrypted SNPs can be stored either at the SPU or with the patient's. Naturally, we recommend the enforcement of some access control mechanism to protect data from unauthorized access even if encrypted. We also assume adversaries to be semi-honest (aka *Honest-but-Curious*). Finally, we assume that K is stored on a smartcard provided to the user. These are in-line with the assumption in the ARHR13 protocols that also require user to have a smartcard that participates in the protocol execution.

Protocol Sketch. We aim to design a protocol that supports the calculation of the disease susceptibility, in such a way that SNPs are never *in the clear* outside the smartcard, and the trade secrets of the MC are also kept confidential. The protocol is as follows:

- To facilitate computation, the MC encrypts the weights using the AH-ECC cryptosystem introduced in Sec. 2.3. Each weight w_j is encrypted as $E_x[w_j] = W_j = (k_j \cdot G, (x \cdot k_j) \cdot G + w_j \cdot H)$, with k_j fresh secrets and x the private key only known to the MC.
- Whenever a test is to be performed, the MC provides the patient's smartcard with the encrypted weights W_j as well as an encryption of zero ($E[0]$). The smartcard initializes a register R with the a re-randomized encryption of zero, i.e., $R = E[0]^k$, for a random k. Then, in a streaming fashion, for each value of W_j (encrypted weights received from MC) and I_j (received from the SPU or read from local storage), it updates the register R as $R \leftarrow R + I_j \cdot W_j$. Note that because computation is in a streaming fashion, the smartcard only needs to keep O(1) state.
- Since the value of I_j is binary, each step of the computation either involves a single elliptic curve point addition or none. Naturally, only the SNP positions for which W_j weights have been provided have to be considered for accumulation.
- When all weights W_j have been processed their signature is checked to ensure they represent a valid test. The accumulated value R is then output from the smartcard, and sent to the MC for decryption. To decrypt $R = (A, B)$, on input the private key x, the MC computes $H^S = B - x \cdot A$. Since the discrete log problem is assumed to be hard in the elliptic curve group, recovering S requires using precomputed tables of some maximal size of S.

Remarks. Note that, while the patient's smartcard is involved in the protocol, the smartcard does not actually perform any (expensive) elliptic curve point multiplications, since it leverages the binary nature of I_j to only run (cheaper) additions. Modern smartcards can perform thousands of elliptic curve additions a second. The internal state of the smartcard comprises a symmetric key K, a handful of elements, and a couple of accumulated hashes – all operations are performed in a streaming manner as data is received. Also, observe that the MC only needs to store a key, and perform a single decryption per test. The tables used for decryption can be re-used even if the private key is rotated. Finally, note that the potentially large number of encrypted SNPs, as well as the encrypted weights W_j, can be pre-fetched or downloaded ahead of time, and can be stored on untrusted storage devices.

Security. We defer formal security proofs to a future extended version of the paper, but, as mentioned above, the security of the scheme stems, in a straightforward way, from the intractability of the Elliptic Curve Discrete Logarithm Problem (ECDLP).

4. SECRET-SHARING BASED PROTOCOL

We now present our second protocol, which is based on secret-sharing and relies on infrastructure servers to improve on efficiency. Specifically, we propose a protocol that makes use of two distinct parties, assumed not to collude with each other. These two parties can be embodied by the SPU and the MC, which in the setting of the original ARHR13 protocol are also assumed not to be colluding and entrusted with a similar burden of computation.

Protocol Sketch. We assume that the CI produces and provides the patient with a smartcard containing the private key y corresponding to a public key $y \cdot G$, where G is a public point on a secure elliptic curve. The public key is then used to output an El-Gamal encrypted stream of all indicator variables I_j corresponding to the patient's SNPs. Each ciphertext is a pair of elements:

$$U_j = (k_j \cdot G, k_j \cdot y \cdot G + I_j \cdot H),$$

where H is a random public point on the elliptic curve.

The weights w_j are also encoded by the MC as a sequence of random values u_j and v_j under the constraint that $u_j + v_j \equiv w_j \mod q$ (q is the order of the group formed by the elliptic curve). Knowledge of either sequences leaks no information about the secret weights w_j. The sequences u_j and v_j are distributed to SPU and MC, respectively. The test computation proceeds as follows:

- The encrypted SNPs, U_j, are loaded by the SPU and MC.
- The SPU and the MC use the shares of the secret weights to compute the following sums over elliptic curve points:

  $$A = \sum_j u_j \cdot U_j \qquad B = \sum_j v_j \cdot U_j$$

  (Addition here denotes pairwise addition of two elliptic curve points, i.e., the elements of the ciphertext.)
- The ciphertexts A and B are sent to the patient's smartcard and used to compute $D = A + B$, which is decrypted using the secret y inside the smartcard. This yields an element $H^S = H^{\sum_j w_j \cdot I_j}$, and the test result S can be recovered through the use of a lookup table, either at the patient or using a service at the MC.

Remarks. Compared to the first protocol (Sec. 3), this construction offers a few advantages. Since the secret in the smartcard is only used at the end of the protocol, disease susceptibility tests may be pre-computed once the tests are designed, and then decrypted when the smartcard is provided by the user as a way of authorizing result disclosure. Until that point, no genetic information is leaked. Also, note that the scheme may be generalized to any number of trusted parties and shares, or collapsed into a simpler computation if, e.g., a pharmaceutical company acts as a trusted service to facilitate the computation. Note that both the SPU and the MC have to perform a number of (full) elliptic curve multiplications to compute the cipher texts A and B, but the smart card only needs to perform two additions and a single decryption to uncover the result.

Security. While we defer formal security proofs to a future extended version of the paper, it is easy to observe that the privacy of the weights is guaranteed by the security of AH-ECC (secure under the ECDLP assumption), and the privacy of the SNPs is based on the security of the conventional, bulk, encryption scheme.

5. PERFORMANCE COMPARISON

We now analyze the performance of our proposed PDS protocols, and compare them to the ARHR13 [3] scheme.

Pre-processing. We start with analyzing the complexity of the SNPs' encryption. According to [13], there are approximately 50 million

known SNPs. While any individual carries (on average) about 4 million SNPs, both ARHR13 and our new protocols require that the CI encrypts all of patient's SNPs (i.e., both real and potential SNPs). Obviously, this operation is performed only once, at sequencing time, so we only focus on the *storage* complexity. In ARHR13, for each SNP, two BCP [5] ciphertexts (one for the SNP, one for the SNP's square) are sent to, and stored at, the SPU. Considering a 112-bit security parameter, each BCP ciphertext is a pair of 4096-bit group elements. Therefore, the encryption of 50M SNPs generates $2 \cdot (5 \cdot 10^7) \cdot (2 \cdot 4096)$ bits, i.e., almost 100GB. On the contrary, the encryption of either all the SNPs or all the weights with our proposed schemes takes $2 \cdot (5 \cdot 10^7) \cdot (2 \cdot 193)$ bits, i.e., about 4.5GB, which represents an order of magnitude improvement.

PDS Testing (Computation). Next, we review the efficiency of the different PDS protocols, focusing on the online phase. In ARHR13, the computational complexity is dominated by the homomorphic operations at the MC, involving the encrypted SNPs and weights/ contributions. This requires multiplying some encrypted integers to some constants – see Eq. 6 in [3]. In the ciphertext domain, these operations correspond to modular exponentiations, specifically, for each SNP, the MC needs to perform three such multiplications, hence, six BCP modular exponentiations. Exponents are drawn from a small group, since, as explained in [3], these exponents are probabilities multiplied by 1000. To estimate protocol's running time, we implemented the BCP cryptosystem in C, using OpenSSL, and measured the time to perform the homomorphic operations on a Macbook Air equipped with a 1.7 GHz Intel Core i7 CPU. Over 1000 runs, we measured an average of 3.5ms per SNP.

As discussed in Sec. 1, in order to actually hide the identity and the size of SNPs contributing to the disease susceptibility, the test should involve all possible "interesting" SNPs. As we were told by genetics experts, this is in the order of 1M. Thus, when considering 1M SNPs, ARHR13 incurs, on the MC, running times in the order of 3.5ms $\times$ 1M, i.e., approximately 1 hour.

We also implemented, in C/OpenSSL, our two new protocols and measured, on the same machine, the running times to execute the test. For our first, smartcard-based protocol, we measured 13 mins on the MC to encrypt 1M SNPs (which can actually be precomputed ahead of time) and 7s for the operations on the patient's side. Note that the user-side computation is assumed to run on a smartcard, thus, running times will likely be 1 order of magnitude longer than 7s, but this would still be significantly more efficient than ARHR13. Also, recall that only (very efficient) point additions are executed on the patient's smartcard. For our second, secret sharing-based protocol, it took 14 mins to run the aggregation protocol between the SPU and the MC with 1M SNPs. Note that this can be executed offline, without the patient's involvement (the smartcard is only needed to decrypt the final result). Therefore, the online time really depends on a single smartcard operation, independently from the number of tested SNPs (thus, it is in the order of milliseconds).

PDS Testing (Communication). Finally, we measure the traffic generated by each PDS protocol's online phase. With ARHR13, the SPU needs to send to the MC two BCP [5] ciphertexts for each SNP (the SNP encryption and its squares). Assuming 1M SNPs, this entails $2 \cdot 10^6 \cdot (2 \cdot 4096)$ bits, i.e., roughly 2GB. Whereas, the communication complexity of our first (smartcard-based) protocol is dominated by the transfer of the encrypted weights from the MC to the patient, which amounts to 92MB in our experiments. The second protocol is very economical in terms of bandwidth on the user device. However, the CI still needs to send, to the SPU and the MC, 92MB each, even though this can be done before the tests take place, and amortized over multiple tests.

6. CONCLUSION

This short paper presented two protocols for private disease susceptibility tests, following the model proposed in ARHR13 [3]. We started by pointing out a few limitations of ARHR13, specifically, the fact that the protocol leaks which and how many SNPs are being tested, potentially allowing an adversary to infer information about the test's specifics and/or the disease the patient is being tested for.

We introduced an alternative (but equivalent) encoding of the SNPs, which allows us to reduce to the problem of summing products of secrets. This can either be done by relying on a smartcard at the patient's side, or using secret sharing on infrastructure servers. Both protocols ensure that genetic information is always encrypted when in transit or on general purpose hardware, and use a secret within a smartcard to unlock the result of the test. Since the numerical domain of the test results is small, we used the simpler and more efficient/compact AH-ECC scheme, as opposed to the expensive BCP cryptosystem [5]. This means that our protocols can scale and be applied to a very large number of SNPs, thus hiding the nature of the test performed and protecting the trade secrets involved.

As part of future work, we plan to release a more detailed description of the protocols, a deployment on smartcards, and a full open-source implementation.[1] Finally, we plan to adapt our protocols to other settings where linear models are used, e.g., adjusting Warfarin drug dosing based on genetic traits.

Acknowledgments. We thank the authors of [3] for their valuable feedback and for sharing the source code of their implementations.

References

[1] E. Ayday, E. De Cristofaro, J.-P. Hubaux, and G. Tsudik. The Chills and Thrills of Whole Genome Sequencing. *IEEE Computer*, 2014.

[2] E. Ayday, J. L. Raisaro, U. Hengartner, A. Molyneaux, and J.-P. Hubaux. Privacy-preserving processing of raw genomic data. In *DPM*, 2013.

[3] E. Ayday, J. L. Raisaro, J.-P. Hubaux, and J. Rougemont. Protecting and Evaluating Genomic Privacy in Medical Tests and Personalized Medicine. In *WPES*, 2013.

[4] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik. Countering GATTACA: Efficient and Secure Testing of Fully-Sequenced Human Genomes. In *CCS*, 2011.

[5] E. Bresson, D. Catalano, and D. Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In *ASIACRYPT*, 2003.

[6] Y. Chen, B. Peng, X. Wang, and H. Tang. Large-Scale Privacy-Preserving Mapping of Human Genomic Sequences on Hybrid Clouds. In *NDSS*, 2012.

[7] E. De Cristofaro, S. Faber, P. Gasti, and G. Tsudik. GenoDroid: Are Privacy-Preserving Genomic Tests Ready for Prime Time? In *WPES*, 2012.

[8] E. De Cristofaro, S. Faber, and G. Tsudik. Secure Genomic Testing with Size-and Position-Hiding Private Substring Matching. In *WPES*, 2013.

[9] Y. Erlich and A. Narayanan. Routes for breaching and protecting genetic privacy. *Nature Reviews Genetics*, 15(6), 2014.

[10] G. Ginsburg and H. Willard. Genomic and Personalized Medicine: Foundations and Applications. *Translational Research*, 154(6), 2009.

[11] 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.

[12] A. Johnson and V. Shmatikov. Privacy-preserving data exploration in genome-wide association studies. In *KDD*, 2013.

[13] National Center for Biotechnology Information (US). Single Nucleotide Polymorphism Database. http://www.ncbi.nlm.nih.gov/projects/SNP/.

[14] R. Wang, X. Wang, Z. Li, H. Tang, M. Reiter, and Z. Dong. Privacy-preserving genomic computation through program specialization. In *CCS*, 2009.

[1] Preliminary code is already available at: https://github.com/gdanezis/genepriv.