SECURE GENOMIC SUSCEPTIBILITY TESTING BASED ON LATTICE ENCRYPTION

Juan Ramón Troncoso-Pastoriza, Alberto Pedrouzo-Ulloa, Fernando Pérez-González

Signal Theory and Communications Department
University of Vigo, 36310 Vigo, Spain
{troncoso,apedrouzo,fperez}@gts.uvigo.es

ABSTRACT

Recent advances in Next Generation Sequencing have increased the availability of genomic data for more accurate analyses, like testing for the genetic susceptibility to a disease. Current laboratories' facilities cannot cope with this data growth, and genomic processing needs to be outsourced, comprising serious privacy risks. This work proposes an encrypted genomic susceptibility test protocol based on lattice homomorphic cryptosystems, and introduces optimizations like data packing and transformed processing to achieve considerable gains in performance, bandwidth and storage needs.

Index Terms— Genomic Privacy, Lattice-Based Cryptography, Homomorphic Encryption, Privacy Protection

1. INTRODUCTION

Genomic research has experienced a considerable growth in the last years due to the advances in Next Generation Sequencing (NGS), which enable potentially better analyses, tests, diagnostics and treatments based on genomic data. The growing volume of genomic data available to be processed, cannot be managed by current facilities at hospitals and laboratories. The need for outsourced genomic processing is urgent, but it entails severe privacy risks [1] comprising, among others, re-identification threats (it is not possible to entirely anonymize genomic data), phenotype inference (sharing aggregate genomic data, even pseudonymized, enables kin privacy breaches), and other threats (anonymous paternity breaches, legal and forensic inferences), affecting not only the individual but also his/her ancestors and descendants.

Several proposals of privacy-preserving mechanisms have arisen to cope with these threats in two main fields: research studies like Genome-Wide Association Studies (GWAS), and personalized health-care. While the former has been recently tackled through differentially-private mechanisms [2, 3], dealing with person-level genome sequence records prevents the use of generalization techniques or differentially-private mechanisms, and the solution must involve cryptographic primitives, which are generally costlier than other approaches.

One of the most recent privacy-preserving mechanisms for disease susceptibility outsourced processing was proposed by Ayday *et al.* [4], which introduce an untrustworthy Storage and Processing Unit (SPU) to deal with the outsourced encrypted processing, and devise a protocol based on additive homomorphic encryption and proxy decryption to enable the calculation of simple susceptibility tests on a set of Single Nucleotide Polymorphisms (SNPs) of one patient; this encrypted test is eventually handled by the medical center due to the limitations of the used homomorphism. Subsequently, Namazi *et al.* [5] proposed the use of lattice-based somewhat homomorphic encryption (SHE) to move the computation complexity to the SPU, but they did not evaluate it nor addressed the shortcomings introduced when dealing with SHE, namely increased cipher expansion, higher bandwidth requirements and much higher storage needs for the encrypted sequences. In this work, we propose an efficient protocol to deal with encrypted genomic susceptibility tests based on Ring Learning with Errors (RLWE) cryptosystems, and introduce optimizations which lead to a considerable improvement in terms of computation, bandwidth and storage with respect to both the original protocol by Ayday *et al.* [4] and Namazi *et al.* [5].

Uppercase letters denote matrices and lowercase letters denote elements from a vector space. a_{E_P} denotes the result of the encryption of a with the key belonging to P. The rest of the paper is organized as follows: Section 2 briefly introduces the used cryptosystem and its primitives. Section 3 revisits the scheme by Namazi *et al.* [5]. Section 4 describes our proposed protocol and the introduced optimizations. Section 5 evaluates the secure protocol in terms of ciphertext size, run times and communication, and compares it to the prior works.

2. RLWE-BASED SHE

We choose Lauter *et al.*'s [6] as our cryptosystem, due to its simplicity, efficiency and security, but any other RLWE cryptosystem (as FV [7] or BGV [8]) can be used as well. Table 1 summarizes its parameters and primitives.

Furthermore, by means of a relinearization matrix $\boldsymbol{B}$ it is possible to transform three-component encryptions after a ho-

This work was partially funded by the Spanish Ministry of Economy and Competitiveness and the ERDF under projects COMPASS (TEC2013-47020-C2-1-R) and COMONSENS (TEC2015-69648-REDC) and the FPI grant (BES-2014-069018), by the Galician Regional Government and ERDF under projects "Consolidation of Research Units" (GRC2013/009), and AtlantTIC, and by the EU H2020 Framework Programme under project WITDOM (project no. 644371).

Table 1. RLWE-based Lauter Cryptosystem

Parameters		
Let $R_t[z] = \mathbb{Z}_t[z]/(z^n+1)$ be the cleartext ring and $R_q[z] = \mathbb{Z}_q[z]/(z^n+1)$ the ciphertext's. The noise distribution $\chi[z]$ in $R_q[z]$ takes its coefficients from a spherically-symmetric truncated i.i.d Gaussian $\mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$; q is a prime $q \equiv 1 \bmod 2n$, and $t < q$ is relatively prime to q.		
Cryptographic Primitives		
SH.KeyGen	Process	$s, e \leftarrow \chi[z]$, $a_1 \leftarrow R_q[z]$ $sk = s$ and $pk = (a_0 = -(a_1 s + te), a_1)$
SH.Enc	Input	$pk = (a_0, a_1)$ and $m \in R_t[z]$
	Process	$u, f, g \leftarrow \chi[z]$ and the fresh ciphertext is $\boldsymbol{c} = (c_0, c_1) = (a_0 u + tg + m, a_1 u + tf)$
SH.Dec	Input	sk and $\boldsymbol{c} = (c_0, c_1, \ldots, c_{\gamma-1})$
	Process	$m = \left(\left(\sum_{i=0}^{\gamma-1} c_i s^i\right) \bmod q\right) \bmod t$
SH.Add	Input	$\boldsymbol{c}_0 = (c_0, \ldots, c_{\beta-1})$ and $\boldsymbol{c}_1 = (c'_0, \ldots, c'_{\gamma-1})$
	Process	$\boldsymbol{c}_{add} = (c_0 + c'_0, \ldots, c_{\max(\beta,\gamma)-1} + c'_{\max(\beta,\gamma)-1})$
SH.Mult	Input	$\boldsymbol{c}_0 = (c_0, \ldots, c_{\beta-1})$ and $\boldsymbol{c}_1 = (c'_0, \ldots, c'_{\gamma-1})$
	Process	Using a symbolic variable v their product is $\left(\sum_{i=0}^{\beta-1} c_i v^i\right) \cdot \left(\sum_{i=0}^{\gamma-1} c'_i v^i\right) = \sum_{i=0}^{\beta+\gamma-2} c''_i v^i$

momorphic product back into two-component fresh-like encryptions. This matrix can also be used as a proxy reencryption in order to perform the key change needed at the end of the protocol (see Section 3). In this case, the relinearization process of a multiplied ciphertext (vector of 3 components in R_q) $\boldsymbol{c}_{P_1} = \{c_1, c_2, c_3\}$ under P_1's key into P_2's key can be expressed as a matrix product $\boldsymbol{c}_{P_2} = \{c_1, c_2\} + \boldsymbol{c_{3,base-t}} \cdot \boldsymbol{B}$, where $\boldsymbol{c_{3,base-t}}$ is a $\lceil \log_t q \rceil$-length row vector with the base-t decomposition of the polynomial c_3, and matrix $\boldsymbol{B}$ has size $\lceil \log_t q \rceil \times 2$ (see [9] for further details).

The equivalent bit-security of this cryptosystem can be lower-bounded by $t_{BKZ}(\delta) = \frac{1.8}{\log_2 \delta} - 110$ [10], where δ is the Root Hermite Factor of the used polynomial lattice.

3. ENCRYPTED SUSCEPTIBILITY TESTS

The genomic sequence of each individual presents variations with respect to the reference sequence which fully identify the individual. The most common and relevant variants are called SNPs (Single Nucleotide Polymorphisms), which are particularly suitable for running susceptibility tests of certain diseases. Weighted averaging [11] is the simplest way to measure the susceptibility of a patient P to a disease x:

$$S^{P,x} = \sum_{i \in \Omega_x} \overline{c}^{x,i} \{ pr_0^{x,i} [1 - \text{SNP}^{P,i}] + pr_1^{x,i} [\text{SNP}^{P,i}] \}. \quad (1)$$

The symbols used in Eq. (1) are defined in Table 2. As this test involves a bounded number of additions and products, an SHE scheme allows to execute it with all the inputs encrypted.

Table 2. Used Notation

Γ^P	Set of positions of real SNPs of patient $\mathcal{P}$
γ^P	Set of positions of potential SNPs of patient $\mathcal{P}$
$\text{SNP}^{P,i}$	i-th SNP for patient $\mathcal{P}$. $\text{SNP}^{P,i}$ equals 0 when it belongs to γ^P, and 1 when the patient presents a variant (it belongs to Γ^P)
Ω_x	Set of *relevant* positions of SNPs which are related to disease x.
$pr_b^{x,i}$	$\Pr(x \vert SNP^{P,i} = b)$, with $b \in 0, 1$. Probability of developing disease x conditioned on the value of the i-th SNP
$\overline{c}^{x,i}$	Normalized contribution of $SNP^{P,i}$ to the susceptibility to x.
$S^{P,x}$	Predicted susceptibility of patient $\mathcal{P}$ to disease x

We briefly revisit the protocol by Namazi et al. [5] to calculate Eq. (1) homomorphically, with the following parties: a patient P owns a biological sample; a medical center MC has the knowledge of the parameters (pr, c) for calculating the susceptibility to disease x; the certified institution CI is a trusted party that sequences the patient's DNA and generates all the used cryptographic keys; the Storage and Processing Unit SPU is an untrustworthy party with computational power to execute the encrypted test. The patient does not trust the MC to share all his/her genomic data, and both MC and P distrust SPU with respect to the analysis parameters and the patient's data. All parties are considered to be semi-honest. The protocol works as follows (see Figure 1):

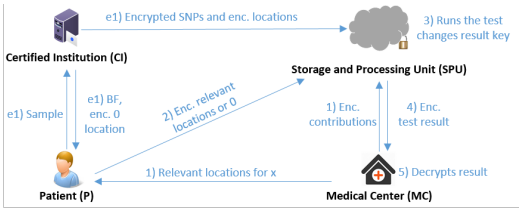

Fig. 1. Encrypted susceptibility testing protocol.

Step s1: The CI generates and distributes the needed keys: P and MC have one SHE key-pair each, while P and CI share a symmetric key $sk_{P,CI}$; the CI also produces a relinearization matrix $\boldsymbol{B}$ to change encryptions from P's key into MC key, and sends it to the SPU.

Sequencing and generation of input encryptions

Step e1: After P sends the biological sample to CI, the latter sequences it, builds a Bloom Filter representing the positions for which the patient presents SNPs, and sends it to P; CI encrypts these positions $\{l_{i,E_{P,CI}}\}$ and a "zero position" $l_{0,E_{P,CI}}$ with $sk_{P,CI}$, and the values of all SNPs $\text{SNP}^{P,i}$ with P's SHE key, and sends all these encryptions to the SPU.

Encrypted susceptibility test

Step 1: The MC marks the location of SNPs in Ω_x and sends them to P. Additionally, it sends the contributions of these SNPs to the disease x encrypted under P's SHE key to SPU: $\{[pr_b^{x,i} \cdot \overline{c}^{x,i}]_{E_P}\}_{b \in \{0,1\}, i \in \Omega_x}$.

Step 2: P runs the Bloom filter for these positions; for those in the filter (present variants), P encrypts the corresponding location $l_{i,E_{P,CI}}$ and sends it to SPU; otherwise, P sends the encryption $l_{0,E_{P,CI}}$.

Step 3: The SPU computes the susceptibility Eq. (1) on patient's encrypted SNPs and MC's encrypted susceptibility parameters for x by using the homomorphic properties of the SHE scheme, obtaining the encryption of $S_{E_P}^{P,x}$ under P's key.

Step 4: The SPU uses the relinearization matrix to switch the result into MC's key, and sends it to MC.

Step 5: The MC decrypts the clear-text test result $S^{P,x}$ of patient $\mathcal{P}$ for the disease x using its own SHE secret key.

This protocol succeeds in moving all homomorphic computation to the SPU and keeping the locations and values of P's SNPs concealed from the SPU and the MC, and the test parameters concealed from the SPU. Conversely, its high cipher expansion makes it much more demanding in terms of

storage and bandwidth compared to the Paillier based scheme by Ayday *et al.*, as we show in Section 5.

4. PROPOSED APPROACH

As can be seen from the protocol description in Section 3, the only elements which have to be encrypted with a homomorphic encryption are the patient SNPs, and the susceptibility parameters; Ayday *et al.* [4] encrypted only the patient SNPs, as the computation was done at the MC, which already knows the clear-text susceptibility contributions. Blindly applying lattice encryptions to the protocol produces a huge growth in the cipher expansion: SNPs are binary values (either present 1 or absent 0), which get encrypted into several thousand bits in Paillier, and several hundred thousand bits with an RLWE cryptosystem. Hence, even when the lattice-based operations are more efficient than their Paillier-based counterparts, the large cipher expansion becomes a serious drawback when coping with 4 million SNPs per patient.

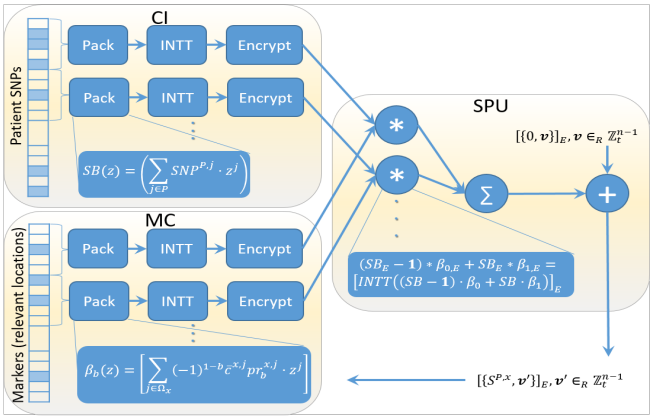

Fig. 2. Diagram of the encrypted susceptibility computation.

Figure 2 presents a high-level view of our proposed approach for dealing with the encrypted calculation of the susceptibility. We present four main contributions described in the following paragraphs: a judicious choice of the cryptosystem parameters to optimize the performance and maximize the security of the protocol; an input packing strategy to minimize storage and bandwidth; a pre-processing mechanism based on transformed coefficients to enable the homomorphic calculation of component-wise products between vectors of susceptibility coefficients and SNPs, and a homomorphic blinding strategy to enable the seamless calculation of the addition of all the components in one vector while avoiding costly unpacking/repacking operations at the SPU.

4.1. Parameter choice

RLWE cryptosystems work with polynomials in R_q; i.e., the ring product is a polynomial product (convolution). In order to speed up products, it is more convenient to work in a transformed domain with the convolution property, where convolutions become much more efficient component-wise products. As these cryptosystems work in finite rings, we stick to Number Theoretic Transforms (NTTs) instead of Discrete Fourier Transforms (DFTs), which would introduce undesirable rounding errors [9]. For an n-th root of unity α in the ring, the NTT has a similar form to the DFT:

$$NTT\{x\} = \sum_{i=0}^{n-1} x[i] \cdot \alpha^{ik},\ INTT\{X\} = n^{-1} \cdot \sum_{k=0}^{n-1} X[k] \cdot \alpha^{-ik}.$$

Therefore, we parameterize the cryptosystem to enable component-wise operations in the NTT domain. We choose $n = 2^k$ (polynomial degree in R_q) as a power of 2, and q and t as Proth primes $(c \cdot 2^k + 1)$ [9]; this choice guarantees that an n-th root of unity exists in $\mathbb{Z}_q$ (ciphertext coefficients) and in $\mathbb{Z}_t$ (plaintext coefficients), in such a way that NTTs of size n exist both in $\mathbb{Z}_q$ and $\mathbb{Z}_t$. All the used polynomials (random polynomials, input plaintexts and keys) undergo an NTT prior to encryption, all ciphertexts are always expressed in the NTT domain, and decryptions are followed by an INTT of the resulting polynomial.[1] Hence, all the intermediate operations are considerably faster (component-wise), and encryption and decryption suffer from a slight overhead for calculating the NTT/INTT with fast algorithms ($O(n \log(n))$).

4.2. Input Packing

Due to the polynomial structure of RLWE cryptosystems, the cipher expansion can be reduced by packing the inputs in vectors of n elements (as many as the degree of the polynomials in R_q, see Table 1) instead of encrypting one scalar value per ciphertext. For the devised susceptibility test protocol, the CI can encrypt the SNPs of the patient in blocks of n SNPs per ciphertext, which divides the storage overhead by a factor of n. This creates a two-level indexing of the SNPs (i, j), where i indexes the block where the SNP was encrypted, and j indexes the polynomial coefficient ($j \in \{0, n-1\}$) where the SNP was packed inside the block. The mapping between the SNP location and the indices (i, j) can be freely chosen by the CI, and must be known by the MC. This alters steps 1 and 2 of the protocol: In step 1, the MC encrypts the contributions of a SNP indexed by (i, j) in the j-th coefficient of the polynomial, and zeros in the other coefficients. If several relevant SNPs belong to the same block, their contributions are packed together in the same encryption. In step 2, after running the Bloom Filter, P sends to the SPU the encrypted location $l_{i,E_{P,CI}}$ indexing the chunks of SNPs where the relevant positions belong, and sends no information about j.

4.3. Packed operations: pre-processing

Once the inputs are packed, the calculation of Eq. (1) requires the homomorphic execution of component-wise products of SNP contributions and SNP values. This is not possible if we

[1] In order to perform cyclic convolutions inside a negacyclic ring (mod $x^n + 1$), signals must be pre- and post-processed with a component-wise product with a vector of powers of a root of -1 in $\mathbb{Z}_q$ [9]. This operation is already accounted for in all the measured run times.

Table 3. Evaluation runtimes, bandwidth and storage for 4M SNPs and a 10-marker test ($|\Omega_x| = 10$).

Run time [ms] / transferred size		CI	SPU		MC	
Ayday *et al.*[4]		Encrypt per SNP	Recrypt	Proxy recrypt	Homomorphic calc.	Paillier decrypt
2048 bit modulus, 112 bit security		33,2 ms / 4,1 GB	304,3 ms / 10,2 kB	30,3 ms / 1,02 kB	39,3 ms / 1,02 kB	30,3 ms / —
		Encrypt per SNP	Homomorphic calc.	Relinearization	Encrypt params	RLWE Decrypt
RLWE $n = 4096$	Unpacked	0,45 ms / 262,1 GB	2,17 ms / —	2,32 ms / 65,5 kB	9,1 ms / 1,31 MB	0,96 ms / —
364 bit security ($\delta = 1.002$)	Packed	0,00011 ms / 64 MB	0,1 – 2,17 ms / —	2,32 ms / 65,5 kB	0,45 – 9,1 ms / 0,131 – 1,31 MB	0,96 ms / —
RLWE $n = 2048$	Unpacked	0,22 ms / 131,1 GB	1,08 ms / —	1,1 ms / 32,8 kB	4,5 ms / 655 kB	0,46 ms / —
127 bit security ($\delta = 1.005$)	Packed	0,00011 ms / 64 MB	0,05 – 1,08 ms / —	1,1 ms / 32,8 kB	0,22 – 4,5 ms / 65,5 – 655 kB	0,46 ms / —

encrypt the input blocks of SNPs directly, as the cryptosystem only allows for homomorphic convolutions. Hence, the CI (resp. MC) first applies an INTT to the polynomial of SNP values (resp. contributions), and then encrypts the transformed values. Then, due to the convolution property of the NTT, the homomorphic operations become:

$$INTT(\{SNP^{P,(i,j)}\}_j) \circledast INTT(\{pr_b^{x,(i,j)}\overline{c}^{x,(i,j)}\}_j) = INTT(\{SNP^{P,(i,j)} \cdot pr_b^{x,(i,j)}\overline{c}^{x,(i,j)}\}_j).$$

These transforms are enabled by our choice of t and n, that guarantees that the n-size INTTs exist for coefficients in $\mathbb{Z}_t$. Therefore, the SPU can seamlessly obtain the encrypted component-wise products contributing to the susceptibility.

4.4. Obtaining the test result

After the previous process, the SPU ends up with an encrypted vector holding the INTT coefficients of the component-wise products, but the cryptosystem homomorphism does not allow to add them together without decrypting and unpacking them first. To overcome this limitation, we leverage the structure of the NTT, by realizing that the first coefficient of the INTT is just the sum of all the signal coefficients in the time domain, multiplied by the modular inverse of n in $\mathbb{Z}_t$. Hence, the SPU generates a random vector $\boldsymbol{v} \in \mathbb{Z}_t^{n-1}$ to blind the remaining INTT coefficients, and homomorphically adds it to the packed susceptibility encryption (at the end of step 3). Then, after performing the relinearization and sending back the resulting encryption to MC (step 4), the latter can decrypt the result and obtain a vector which holds the susceptibility result $S^{P,x}$ in the first coefficient (multiplied by $n^{-1} \mod t$) and random values in the remaining coefficients. Hence, we also avoid that the MC has to execute an NTT to revert the INTT that was applied to the inputs.

5. IMPLEMENTATION AND EVALUATION

We implemented the full protocol in C++ with and without packing, using the NFLlib [12] library, and Ayday's Paillier-based version with GMP [13]. According to Section 4.1, we choose $t = 65537$, as it is enough to deal with all the input values with a precision of 10^{-3} for a test of up to 65 markers; due to efficiency reasons, we fix q to 62 bits, such that it fits in a limb (8 bytes) and all operations on polynomial coefficients are performed in just one machine cycle; additionally, this choice of q and t allows for the correct computation of one encrypted polynomial product between two fresh encryptions, which is enough to homomorphically calculate Eq. (1).

We choose medium-term security for Paillier, with 2048-bit modulus (112 bits of security), and two levels of security for our lattice-based protocol: $n = 2048$, which produces an equivalent security of 127 bits ($\delta = 1.005$, see Section 2), and $n = 4096$, with 364 bits of security ($\delta = 1.002$). Table 3 shows the run times for each party on an Intel Core i5-2500 processor at 3.3 GHz running Linux, and the sizes of the transferred encryptions at each step for 4 million SNPs per patient and a test with 10 relevant SNPs (markers) in Ω_x.

The RLWE-based protocols considerably outperform the Paillier-based Ayday *et al.* protocol in terms of efficiency (two orders of magnitude for SPU and CI, and one order of magnitude for the MC), while keeping all the homomorphic computation at the SPU instead of the MC. As for the bandwidth, the unpacked solution suffers from the big cipher expansion of the RLWE encryptions, producing a huge set of encrypted SNPs at the CI. The proposed strategies greatly reduce this overhead, limiting the stream of the 4 million encrypted SNPs to just 64 MB, notably lower than the 4 GB needed for the Paillier encryptions, improving on storage needs. The improvement achieved on homomorphic computation depends on the number of blocks spanned by the positions of the relevant SNPs, analogously to the bandwidth needed between SPU and CI. Both can be optimized by configuring the (public) ordering of the SNPs (mapping of the indices (i, j)) so that most of the SNPs relevant for the same diseases be together in the same block.

It must be noted that the performed packing, the used SNP indexing and the blinding of the resulting vector leak no information either to the SPU or to the MC, in such a way that the same security properties and privacy guarantees of the unpacked Paillier-based protocol are preserved here.

6. CONCLUSIONS

We propose a privacy-preserving genomic susceptibility protocol based on a Ring Learning with Errors SHE cryptosystem which outperforms previous protocols in terms of efficiency, bandwidth and storage needs. We introduce a choice of cryptosystem parameters to optimize the performance and the security of the protocol, and propose a transformed input packing strategy to minimize storage and bandwidth, and enable the homomorphic calculation of the susceptibility function while avoiding costly unpacking/repacking operations.

7. REFERENCES

[1] Muhammad Naveed, Erman Ayday, Ellen W Clayton, Jacques Fellay, Carl A Gunter, Jean-Pierre Hubaux, Bradley A Malin, and XiaoFeng Wang, "Privacy in the Genomic Era," *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, pp. 6, 2015.

[2] Aaron Johnson and Vitaly Shmatikov, "Privacy-Preserving Data Exploration in Genome-Wide Association Studies," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, Sep. 2013, KDD '13, pp. 1079–1087.

[3] Fei Yu, Stephen E. Fienberg, Aleksandra B. Slavkovi, and Caroline Uhler, "Scalable privacy-preserving data sharing methodology for genome-wide association studies," *Journal of Biomedical Informatics*, vol. 50, pp. 133 – 141, 2014, Special Issue on Informatics Methods in Medical Privacy.

[4] E. Ayday, J.L. Raisaro, and J.P. Hubaux, "Privacy-Enhancing Technologies for Medical Tests Using Genomic Data," in *20th Annual Network & Distributed System Security Symposium NDSS*, San Diego, CA, USA, Feb. 2013.

[5] Mina Namazi, Juan Ramón Troncoso-Pastoriza, and Fernando Pérez-González, "Dynamic Privacy-Preserving Genomic Susceptibility Testing," in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*. 2016, IH&MMSec '16, pp. 45–50, ACM.

[6] K. Lauter, M. Naehrig, and V. Vaikuntanathan, "Can Homomorphic Encryption be Practical?," Cryptology ePrint Archive, Report 2011/405, 2011, `http://eprint.iacr.org/`.

[7] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption," Cryptology ePrint Archive, Report 2012/144, 2012, `http://eprint.iacr.org/`.

[8] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) Fully Homomorphic Encryption without Bootstrapping," *ACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 13:1–13:36, July 2014.

[9] Alberto Pedrouzo-Ulloa, Juan Ramón Troncoso-Pastoriza, and Fernando Pérez-González, "Number Theoretic Transforms for Secure Signal Processing," *IEEE Transactions on Information Forensics and Security*, 2017, To appear, `https://doi.org/10.1109/TIFS.2016.2647223`.

[10] R. Lindner and C. Peikert, "Better Key Sizes (and Attacks) for LWE-based Encryption," in *CT-RSA'11*. 2011, pp. 319–339, Springer.

[11] Sekar Kathiresan, Olle Melander, Dragi Anevski, Candace Guiducci, Noël P. Burtt, Charlotta Roos, Joel N. Hirschhorn, Göran Berglund, Bo Hedblad, Leif Groop, David M. Altshuler, Christopher Cheh, and Marju Orho-Melander, "Polymorphisms Associated with Cholesterol and Risk of Cardiovascular Events," *New England Journal of Medicine*, vol. 358, no. 12, pp. 1240–1249, 2008.

[12] Carlos Aguilar-Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancrède Lepoint, "NFLlib: NTT-based Fast Lattice Library," in *Topics in Cryptology - CT-RSA 2016*. Feb. 2016, vol. 9610 of *LNCS*, pp. 341–356, Springer.

[13] "GNU MultiPrecision Library," Online, `https://gmplib.org/`.