Testing for Genetic Diseases on Encrypted Genomes

Secure Searching of Biomarkers Using Hybrid GSW Encryption Scheme

Jung Hee Cheon & Miran Kim & Yongsoo Song
Department of Mathematical Sciences, Seoul National University
jhcheon@snu.ac.kr & alfks500@snu.ac.kr & lucius05@snu.ac.kr

Abstract

The rapid development of genome sequencing technology requires to keep genomic data secure even when stored in the cloud and still used for research. The aim of this work is to create a homomorphic security system for searching a set of biomarkers to encrypted genomes. We suggest an efficient method to securely search and extract the information without complicated computation such as comparison. Our important feature is the encoding of genomic database in the polynomial ring, which is easily annihilated by only one multiplication with the query data. Thus the information of encrypted reference/alternate sequence is contained in the constant term of the output polynomial only if their chromosomes and positions match each other.

Cyclotomic Ring

- $\mathcal{R} = \mathbb{Z}[X]/\Phi_m(X)$ for an integer m (: power of two).
- $n = \phi(m)$ is the degree of $\Phi_m(X)$.
- $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ is the residue ring modulo an integer q.
- For a base B and exponent k s.t. $B^k \geq q$, the word decomposition and power of B of a polynomial in $\mathcal{R}_q$ is defined by

$$WD_B(a) = (a_0, \ldots, a_{k-1}) \in \mathcal{R}_B^k \text{ s.t. } a = \sum_{i=0}^{k-1} B^i \cdot a_i, \quad \mathcal{P}_B(a) = (a, B \cdot a, \ldots, B^{k-1} \cdot a),$$

which satisfies $\langle WD_B(a), \mathcal{P}_B(a) \rangle = a$.

RLWE public key encryption

- Sample a small polynomial s and set the secret key $sk = (1, -s)$.
- Public key pk is generated by sampling a random polynomial a from $\mathcal{R}_q$, a small polynomial e, and returning $pk = (b, a)$ for $b = -as + e$.
- $\vec{c} \leftarrow \mathsf{RLWE.Enc}(m)$: Take a polynomial m in the plaintext space $\mathcal{R}_t$ as an input. Sample a small polynomial v and small polynomials e_1, e_2, and return $\vec{c} \leftarrow v \cdot pk + (\frac{q}{t}m + e_0, e_1)$. Note that an encryption $\vec{c} = (c_0, c_1)$ of m satisfies $\langle \vec{c}, sk \rangle = \frac{q}{t}m + e$ for some small polynomial e.
- A RLWE encryption of $m = m_0 + m_1 X + \cdots + m_{n-1}X^{n-1}$ can be easily converted into a LWE encryption of m_0 of secret vector $\vec{s}$.

GSW symmetric key encryption [GSW13, DM15]

- $C \leftarrow \mathsf{GSW.Enc}(m)$:
  - Take $m \in \mathcal{R}$ as an input.
  - Sample a vector of random polynomials $\vec{a} \in \mathcal{R}_q^{2k}$ and a vector of small polynomials $\vec{e} \in \mathcal{R}^{2k}$.
  - Output the $2k \times 2$ matrix $(\vec{c}_0, \vec{c}_1) = (s \cdot \vec{a} + \vec{e}, \vec{a}) + m \cdot G$

for the Gadget matrix $G = I_2 \otimes \mathcal{P}_B(1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ B^{k-1} & 0 \\ 0 & B^{k-1} \end{bmatrix}$.

Note that an encryption C of m satisfies $C \cdot sk = m \cdot \mathcal{P}_B(sk) + \vec{e}$.

Multiplication of GSW & RLWE ciphertexts [CGGI16]

- GSW ciphertexts *act on* RLWE ciphertexts.

$$\mathsf{Mult} : \{\text{GSW ciphertexts}\} \times \{\text{RLWE ciphertexts}\} \to \{\text{RLWE ciphertexts}\}$$
$$C \in \mathcal{R}_q^{2k \times 2} \quad , \quad \vec{c} = (c_0, c_1) \in \mathcal{R}_q^2 \quad \mapsto \quad WD_B(\vec{c}) \cdot C$$

- If $C \cdot sk = m' \cdot \mathcal{P}_B(sk) + \vec{e}$ and $\langle \vec{c}, sk \rangle = \frac{q}{t}m + e$, then their multiplication satisfies

$$\langle \vec{c}_{\mathsf{mult}}, sk \rangle = (WD_B(\vec{c}) \cdot C) \cdot sk = WD_B(\vec{c}) \cdot (C \cdot sk) = \frac{q}{t}mm' + e^*$$

for $e^* = m'e + \langle WD_B(\vec{c}), \vec{e} \rangle$.

- $\vec{c}_{\mathsf{mult}}$ is a RLWE encryption of mm' with the error e^*.

Encoding Algorithm for Genomic Data

Each variation of database consists of three genomic informations: Chromosome, position, and nucleic acid sequence:

$$\mathsf{Chr}[i] \in \{1, 2, \ldots, 22, X(=23), Y(=24)\}, \quad \mathsf{POS}[i] \in \mathbb{Z}, \quad \text{and} \quad \mathsf{SNPs}[i] \in \{A, T, G, C\}^*,$$

and we use one-to-one data encoding functions:

- Encode $(\mathsf{Chr}[i], \mathsf{POS}[i])$ into $d_i = \mathsf{Chr}[i] + 24 \cdot \mathsf{POS}[i] \in \mathbb{Z}_{2^{32}}$
- Encoding of nucleic acid sequence $\mathsf{SNPs}[i] \mapsto \alpha_i$:
  - Each of SNP is encoded by "$A \mapsto 00, G \mapsto 01, C \mapsto 10, T \mapsto 11$", pad '1' at the end of the string, and fill zeros.
  - Missing (empty) genotype is encoded as the zero string.

Encryption of VCF Files & Query Data

- A VCF file is encoded into the set $\{(d_i, \alpha_i) : 1 \leq i \leq \ell\}$. Construct the polynomial

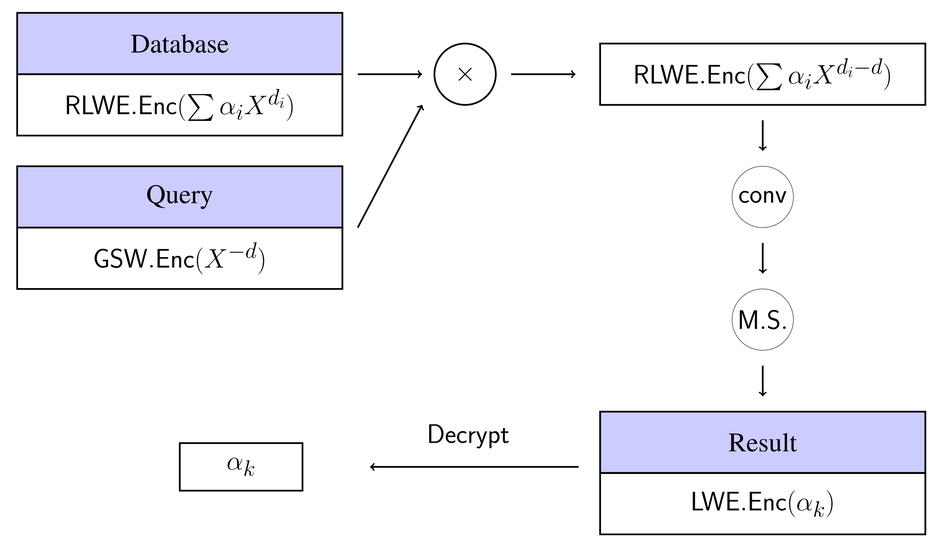
$$\mathsf{SNPs}(X) = \sum_i \alpha_i X^{d_i},$$

and use the public-key RLWE encryption scheme. Store the ciphertext $\vec{c}_{\mathsf{SNPs}}$.

- Use symmetric-key GSW scheme for encoded query (d, α). Encrypt the polynomial $X^{-d} = -X^{n-d}$ and send the ciphertext C_{Q} to the server.
- Goal: check if there is an index $1 \leq j \leq \ell$ such that $(d, \alpha) = (d_j, \alpha_j)$.

Query Computation: Searching and Extraction

- Given $\vec{c}_{\mathsf{SNPs}} \leftarrow \mathsf{RLWE.Enc}(\sum_i \alpha_i X^{d_i})$ and $C_{\mathsf{Q}} \leftarrow \mathsf{GSW.Enc}(X^{-d})$.
- Compute $\vec{c}_{\mathsf{res}} \leftarrow \mathsf{Mult}(C_{\mathsf{Q}}, \vec{c}_{\mathsf{REF}})$, which is an RLWE encryptions of $\sum_i \alpha_i X^{d_i - d}$.
- Convert it into a LWE ciphertext, which is an encryption of α_j if $d_j = d$ for some j; otherwise an encryption of random value.
- Carry out the modulus-switching to reduce communication cost.
- Decrypt the LWE ciphertexts and compare with α.



Optimization Techniques

- Construction of a single polynomial yields huge ring dimension $n > 2^{31}$, so we take $n = 2^{16}$ and divide d_i into two 16-bit integers $d_{i,1}$, $d_{i,2}$.
- The size of the encoded nucleic acid sequences α_i is too large (e.g. 41 bits) to be encrypted in a single ciphertext.
  - Split α_i into smaller integers to use smaller plaintext space $t = 2^{11}$ and modulus $q = 2^{32}$.
  - The use of variable type '*int32_t*' accelerates the speed of implementation and basic C++ std libraries.

Experimental Results

Performance description in terms of complexity & storage with 80-bit security on a single Intel Core i5 running at 2.9 GHz processor.

#(SNPs)	Size	Complexity				Storage	
		Query-enc	DB-enc	Eval	Decrypt	DB	Result
5	10K	0.14s	0.11s	0.67s	0.15ms	1MB	0.25MB
	100K		0.27s	1.64s	0.29ms	2.5MB	0.625MB
20	10K		0.45s	2.75s	0.41ms	4MB	1MB
	100K		1.04s	6.88s	0.84ms	10MB	2.5MB

#(SNPs): maximal number of SNPs considered for comparison

Conclusions

In this work, we suggest an efficient method to securely search of biomarkers using hybrid GSW homomorphic encryption scheme. We came up with a solution to the secure outsourcing matching problem by using polynomial encoding and extraction of query SNP based on the multiplication of a RLWE-GSW ciphertext and an ordinary RLWE ciphertext. Our solution shows the progress of cryptographic techniques in terms of their capability to support real-world genome data analysis.

References

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. to be appeared in ASIACRYPT, 2016.

[DM15] Léo Ducas and Daniele Micciancio. Fhew: Bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology–EUROCRYPT 2015*, pages 617–640. Springer, 2015.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013*, pages 75–92. Springer, 2013.

Acknowledgements

The research was supported by IT R&D program of MSIP/KEIT [No. 0450-21060006].