Private-Key Fully Homomorphic Encryption for Private Classification

Alexander Wood[1,2,3,4], Vladimir Shpilrain[5,6], Kayvan Najarian[2,3,4], Ali Mostashari[7], and Delaram Kahrobaei[1,8]

[1] Department of Computer Science, The Graduate Center, CUNY, USA
[2] Department of Computational Medicine and Bioinformatics, University of Michigan, USA
[3] University of Michigan Center for Integrative Research in Critical Care, USA
[4] Emergency Medicine Department, University of Michigan, USA
[5] Department of Mathematics, The Graduate Center, CUNY, USA
[6] Department of Mathematics, The City College of New York, USA
[7] LifeNome Inc., New York, NY, USA
[8] Department of Computer Science, Tandon School of Engineering, New York University, USA

Abstract. Fully homomophic encryption enables private computation over sensitive data, such as medical data, via potentially quantum-safe primitives. In this extended abstract we provide an overview of an implementation of a private-key fully homomorphic encryption scheme in a protocol for private Naive Bayes classification. This protocol allows a data owner to privately classify her data point without direct access to the learned model. We implement this protocol by performing privacy-preserving classification of breast cancer data as benign or malignant.

Keywords: fully homomorphic encryption, data privacy, machine learning

1 Introduction

Fully-homomorphic encryption (FHE) encompasses potentially quantum-safe primitives which allows for computation of arbitrary functions over encrypted data. The majority of current FHE research is public-key. In contrast, private key cryptosystems require prior knowledge of the encryption/decryption key(s). While this is considered a disadvantage when the goal is purely communication, these cryptosytems are in fact excellent for applications involving sensitive data [20].

A number of papers have approached the problem of private computation over medical data. Some of these applications focus specifically on genomic computation, including edit distance [10], string matching [28, 3], genomic tests such as ancestry and paternity [9], and other genomic tests [23, 22]. Other research focuses on the task of private classification, including neural networks [17, 27], decision trees [5], and Fisher's linear discriminant classifier [19]. All of these applications take place in the public-key setting.

In this extended abstract we propose a fully homomorphic private-key protocol for private Naive Bayes classification, a private argmax protocol, and a new fully homomorphic method of encoding floating point values. We test the protocol on publicly available breast cancer data [24] using the GKS FHE scheme [20] to achieve private classification in under one second.

2 Fully Homomorphic Encryption

A scheme is called additively or multiplicatively homomorphic, respectively, if $[\![x+y]\!] = [\![x]\!] \oplus [\![y]\!]$ and $[\![x \cdot y]\!] = [\![x]\!] \otimes [\![y]\!]$ for operations $\oplus$ and $\otimes$ in the ciphertext space and $[\![\alpha]\!]$ denotes the encryption of a value α. Because a boolean circuit can describe arbitrary comptuation, a scheme is called fully homomorphic if it is both additively and multiplicatively homomorphic [26].

The first FHE scheme was a lattice-based public-key encryption scheme introduced by Gentry which was theoretically revolutionary but impractical implementation-wise [12]. Improvements on this method led quickly to the second generation of fully homomorphic encryption schemes [7, 6, 11] including the BGV scheme [8, 7, 6] and YASHE [4]. More recent improvements have built upon this foundation to yield even faster schemes [2, 16, 13, 15, 14, 21]. These schemes are all lattice-based, which is broadly considered a potentially quantum-resistant primitive.

More recently, interest has grown in private-key fully homomoprhic encryption. The GKS scheme [20] is a ring- and group-based private-key FHE scheme. It avoids some of the computational overhead required for fully homomorphic public key encryption and joins other schemes as a potentially quantum-resistant primitive. The GKS scheme is secure against a ciphertext-only attack.

3 Privacy-Preserving Classification

The main utility of fully homomorphic encryption lies in privacy-preserving classification, where a user classifies her datapoint using a data owner's learned model and neither party learns information about the other party's data [5]. A major application of PPC lies in the medical field. With PPC, a patient could use her medical data to perform medical analyses without worrying about revealing any of her personal information.

Leveled homomorphic encryption schemes (LHE), a variant on fully homomorphic encryption which allows for computation up to a predefined depth, such as YASHE have been used in an application of neural networks to encrypted data called CryptoNets [17]. ML Confidential [19] uses LHE to run classification using Linear Means and Fisher's Linear Discriminant classifiers. The authors in [5] construct protocols for privacy-preserving classification via hyperplane detection, Naive Bayes, and decision trees using two additively homomorphic encryption schemes, Quadratic Reciprocity (QR) [18] and Paillier [25], and one leveled homomorphic encryption scheme, HELib [21].

4 Proposed Method for Fully Homomorphic Private Classification With Naive Bayes

We use private-key fully homomorphic encryption in order to provide private classification using a learned Naive Bayes model. Our method follows from the method of Bost et. al. [5] but varies in several important ways. Bost et. al. re-encode ciphertexts between two additively homomorphic, public-key encryption schemes. Our implementation uses one fully homomorphic, private-key encryption scheme.

Assume that a Data Owner, D, wishes to classify her vector X which contains q features based off of a learned model w owned by a Classification Model Owner, C. The group G contains r distinct classes, $G_1, \ldots, G_r$. During this protocol C should learn no unnecessary information about the input provided by D, and D should learn nothing but the predicted class index of X.

C prepares tables P represented as a column vector of degree r where $P_i = \Pr(G = G_i)$, the prior probability on class G_i, and T, an $r \times q$ matrix where entry T_{ij} represents $\Pr(X = X_j | G = G_i)$. Private classification proceeds as follows:

1: C prepares the tables P and T and sends $[\![P]\!]$ and $[\![T]\!]$ to D.
2: For each class G_i for i from 1 to r, D computes

$$p_i = [\![P_i]\!] \cdot \prod_{j=1}^{p} [\![T_{ij}]\!] = \left[\!\!\left[P_i \cdot \prod_{j=1}^{p} T_{ij} \right]\!\!\right] = [\![\Pr(G_i|X))]\!].$$

3: D computes $i = \underset{1 \le i \le r}{\operatorname{argmax}} [\![\Pr(G_i|X))]\!]$ using a private `argmax` protocol.

The privacy of the learned model is derived from the FHE scheme used during the protocol. The Data Owner's privacy depends on the argmax protocol in step 3. Let $\mathcal{F}$ denote a family of monotone, continuous, additively homomorphic functions that commute with encryption. Our protocol for computing private `argmax` is as follows:

1: Set $I = \{1, 2, \ldots, r\}$.
2: **while** $|I| > 1$ **do**
3: D computes a random permutation π on I and randomly chooses $f \leftarrow \mathcal{F}$
4: D computes $v = f([\![p_{\pi(1)}]\!]) - f([\![p_{\pi(2)}]\!]) = [\![f(p_{\pi(1)} - p_{\pi(2)})]\!]$
5: D sends v to C.
6: C decrypts v and recovers $f(p_{\pi(1)} - p_{\pi(2)})$. If this value is negative, C sends the bit $b = 0$ to D, otherwise send $b = 1$.
7: If $b = 0$, remove $\pi(1)$ from I. Otherwise remove $\pi(2)$.
8: **end while**
9: D returns I.

During this protocol, C collects r values representing the result of a monotone function applied to the difference between random pairs of the posterior probabilities. The application of an unknown monotone function to this difference prevents C from learning partial information from the decrypted value.

5 Implementation

5.1 Fully Homomorphic Encoding

We must encode values in a way which preserves the fully homomorphic properties of the scheme. In the GKS scheme, elements are encoded in the ring

$$S_n = \langle x_1, x_2, \ldots, x_n | p \cdot 1 = 0, x_i^2 = x_i, \\ \text{and } x_i x_j = x_j x_i \text{ for all } i, j \rangle.$$

The plaintext ring P and a public ciphertext ring C are rings of the above form, where $P \subset C$. Elements have coefficients which lie in the field $\mathbb{Z}_p$. The authors of [20] provide a straightforward method of implementing a fully homomorphic encoding of integer values by mapping the element 1 of $\mathbb{Z}_p$ to any idempotent element of S. In order to encode floating point values, the most straightforward approach is to scale the floating point values to integer values with a fixed precision, although decoding requires the user to keep track of the number of times the protocol performs multiplication. However, this approach results in fast overflow over the prime modulus p.

We constructed a more space-efficient encoding for numbers $n \in (-1, 1)$. Each of these values has a base-10 representation $n = (\pm 1) \sum_{i=1}^{\infty} d_i \cdot 10^{-i}$ where d_i is a digit. Let the depth of a digit be the value of its corresponding base-10 exponent i. Any number n as described above can be represented as a collection of (integer, depth) pairs as Encode$(n) = \{(\pm d_i, i) : i \geq 1\}$, where d_i is positive if n is positive and d_i is negative whenever n is negative. Converting an encoded integer back to base 10 is as simple as computing the sum of the digits in base 10.

Add encoded values by adding values which share a depth. To perform multiplication, assume $m = \{(m_i, i) | 1 \leq i \leq B\}$ and $n = \{(n_j, j) | 1 \leq j \leq B\}$ for some maximum depth bound B. Then,

$$m \cdot n = \{(m_i \cdot n_j, i + j) | 1 \leq i \leq B, 1 \leq j \leq B\}.$$

Small values of B provide high accuracy. The value for B in the experiments below is 20 and no loss of accuracy due to depth bound occurs. Observe that while the initial encoding uses integers 0 through 9 in base 10, the integer values after addition or multiplication operations do not necessarily have to be digits. For example, the number .048 would be encoded using the above method as

$$\{(4, 2), (8, 3)\}. \tag{1}$$

However, if we first encoded the numbers 0.6, 0.8, and 0.1 as $\{(6, 1)\}$, $\{(8, 1)\}$, and $\{(1, 1)\}$, then multiplied in their encoded forms we would write

$$0.6 \cdot 0.8 = \{(6, 1)\} \cdot \{(8, 1)\} = \{(48, 2)\}$$

and

$$(0.6 \cdot 0.8) \cdot 0.1 = \{(48, 2)\} \cdot \{(1, 1)\} = \{(48, 3)\}. \tag{2}$$

Observe that the values in equations 1 and 2 both decode to 0.048.

Table 1. Unencrypted Versus Encrypted Experimental Results

Training Set Size	Time (s)		Accuracy		Sensitivity		Specificity	
	Unenc.	Enc.	Unenc.	Enc.	Unenc.	Enc.	Unenc.	Enc.
$S = 100$	0.00001	0.580	0.960	0.946	0.958	0.958	0.974	0.940
$S = 200$	0.00001	0.633	0.962	0.939	0.941	0.941	0.972	0.939
$S = 300$	0.00001	0.631	0.979	0.971	0.949	0.949	0.989	0.978
$S = 400$	0.00001	0.614	0.985	0.968	0.955	0.955	0.995	0.972

5.2 Experiments

To test the above protocols, we implemented a Naive Bayes algorithm to create a learned model and encrypted this learned model using the GKS Encryption Scheme [20]. The size of the ciphertext ring in the experiments was $2^8 = 256$, and the prime modulus was given by $p = 700,000,001$. The family of functions used during private `argmax` is given by

$$\mathcal{F} = \{f : R \to R : f(m) = km\}.$$

for sufficiently small $k \in \mathbb{Z}$ to avoid overflow over the prime modulus p. Our protocols were implemented in C++ and run on a MacBook Pro using El Capitan, a 2.3 GHz Intel Core i7, and with 16 GB memory.

Additive smoothing was performed on the prior probability tables before encryption. Specifically, each probability was increased by 0.1. Any value which was greater than or equal to 1 after smoothing was reset to $0.\overline{9}$, truncated at 20 digits.

Data from the UCI Machine Learning Repository was used to test the performance of the protocols [24]. Specifically, we looked at the Breast Cancer Wisconsin (Original) Data Set which contains 683 complete data points each containing an ID along with 9 attributes and a binary classification. The data gives measurements taken from fine-needle aspirate (FNA) biopsies of benign and malignant breast tumors. Each of the nine attributes was measured by a clinician on a scale of 1 to 10 at the time it was collected. Previous research found that while each measurement holds clinical significance in diagnosing a breast tumor as benign or as malignant, a single attribute is not enough to distinguish between the two cases [29]. In the statistical analysis provided, a positive classification denotes a malignant classification and a negative classification denotes a benign classification.

The time data in the Table 1 represents the number of seconds it takes to classify a single data point. The time increase between encrypted and unencrypted classification is quite steep. However, this is to be expected and occurs to varying degrees in all fully homomorphic implementations.

The loss in accuracy between the encrypted and unencrypted settings occurred due to overflow over the prime modulus p. Observe, however, that this situation was still quite rare, and we were able to attain high accuracy using

only the built-in data types in C++. To increase the size of the prime modulus one could use an arbitrary precision library such as the GNU Multiple Precision Library (GMP) [1] which allows for integer storage above the built-in data type limits in C++.

References

1. The GNU MP Bignum Library, `https://gmplib.org/`
2. Alperin-Sheriff, J., Peikert, C.: Faster Bootstrapping with Polynomial Error. In: Advances in Cryptology CRYPTO 2014. pp. 297–314. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (Aug 2014)
3. Ayday, E., Raisaro, J.L., Hengartner, U., Molyneaux, A., Hubaux, J.P.: Privacy-Preserving Processing of Raw Genomic Data, pp. 133–147. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
4. Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme, pp. 45–64. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
5. Bost, R., Ada Popa, R., Tu, S., Goldwasser, S.: Machine learning classification over encrypted data. Symposium on Network and Distributed System Security (NDSS) (February 2015)
6. Brakerski, Z., Vaikuntanathan, V.: Efficient Fully Homomorphic Encryption from (Standard) LWE. In: Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science. pp. 97–106. FOCS '11, IEEE Computer Society, Washington, DC, USA (2011)
7. Brakerski, Z., Vaikuntanathan, V.: Fully Homomorphic Encryption from ring-LWE and Security for Key Dependent Messages. In: Proceedings of the 31st Annual Conference on Advances in Cryptology. pp. 505–524. CRYPTO'11, Springer-Verlag, Berlin, Heidelberg (2011)
8. Brakerski, Z., Vaikuntanathan, V., Gentry, C.: Fully homomorphic encryption without bootstrapping. In: In Innovations in Theoretical Computer Science (2012)
9. Bruekers, F., Katzenbeisser, S., Kursawe, K., Tuyls, P.: Privacy-preserving matching of dna profiles. Tech. rep. (2008)
10. Cheon, J.H., Kim, M., Lauter, K.: Homomorphic Computation of Edit Distance, pp. 194–212. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
11. Dijk, M.v., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully Homomorphic Encryption over the Integers. In: Advances in Cryptology EUROCRYPT 2010. pp. 24–43. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (May 2010)
12. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing. pp. 169–178. STOC '09, ACM (2009)
13. Gentry, C., Halevi, S., Peikert, C., Smart, N.P.: Ring Switching in BGV-Style Homomorphic Encryption. In: Security and Cryptography for Networks. pp. 19–37. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (Sep 2012)
14. Gentry, C., Halevi, S., Smart, N.P.: Better Bootstrapping in Fully Homomorphic Encryption, pp. 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
15. Gentry, C., Halevi, S., Smart, N.P.: Fully Homomorphic Encryption with Polylog Overhead. In: Advances in Cryptology EUROCRYPT 2012. pp. 465–482. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (Apr 2012)

16. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: CRYPTO. pp. 75–92. Springer (2013)
17. Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., Wernsing, J.: CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. In: PMLR. pp. 201–210 (Jun 2016)
18. Goldwasser, S., Micali, S.: Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information. In: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing. pp. 365–377. STOC '82, ACM, New York, NY, USA (1982)
19. Graepel, T., Lauter, K., Naehrig, M.: ML Confidential: Machine Learning on Encrypted Data, pp. 1–21. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
20. Gribov, A., Kahrobaei, D., Shpilrain, V.: Private-key fully homomorphic encryption in rings. Groups, Complexity, Cryptology **10** (2018)
21. Halevi, S.: HElib: An Implementation of homomorphic encryption (2013), `https://github.com/shaih/HElib`
22. Kim, M., Lauter, K.: Private genome analysis through homomorphic encryption. Cryptology ePrint Archive, Report 2015/965 (2015), `http://eprint.iacr.org/2015/965`
23. Lauter, K., López-Alt, A., Naehrig, M.: Private Computation on Encrypted Genomic Data, pp. 3–27. Springer International Publishing, Cham (2015)
24. Lichman, M.: UCI machine learning repository (2013), `http://archive.ics.uci.edu/ml`
25. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Advances in Cryptology EUROCRYPT 99. pp. 223–238. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (May 1999)
26. Peikert, C.: A Decade of Lattice Cryptography. Foundations and Trends in Theoretical Computer Science **10**(4), 283–424 (Mar 2016)
27. Shokri, R., Shmatikov, V.: Privacy-Preserving Deep Learning. In: Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security. pp. 1310–1321. CCS '15, ACM, New York, NY, USA (2015)
28. Troncoso-Pastoriza, J.R., Katzenbeisser, S., Celik, M.: Privacy preserving error resilient dna searching through oblivious automata. In: ACM Conference on Computer and Communications Security (CCS). p. 519–528. ACM Press, ACM Press, Alexandria, Virginia, USA (Oct 29–Nov 2 2007)
29. Wolberg, W.H., Mangasarian, O.L.: Multisurface method of pattern separation for medical diagnosis applied to breast cytology. Proceedings of the National Academy of Sciences, U.S.A. **87**, 9193–9196 (1990)