Private Naive Bayes Classification of Personal Biomedical Data: Application in Cancer Data Analysis

Alexander Wood, Vladimir Shpilrain, Kayvan Najarian, and Delaram Kahrobaei

***Abstract*—A wealth of data is inaccessible to biomedical researchers and clinicians due to privacy restrictions such as HIPAA. Clinicians would benefit from access to predictive models for diagnosis, such as classification of tumors as malignant or benign, without compromising patients' privacy. In addition, the medical institutions and companies who own these medical information systems wish to keep their models private when in use by outside parties. Fully homomorphic encryption (FHE) enables computation over encrypted medical data while ensuring data privacy. In this paper we use private-key fully homomorphic encryption to design a cryptographic protocol for private Naive Bayes classification. This protocol allows a data owner to privately classify his or her information without direct access to the learned model. We apply this protocol to the task of privacy-preserving classification of breast cancer data as benign or malignant. Our results show that private-key fully homomorphic encryption is able to provide fast and accurate results for privacy-preserving medical classification.**

***Index Terms*—cryptographic protocols, data privacy, fully homomorphic encryption, medical information systems, predictive models**

I. Introduction

CRYPTOGRAPHIC methods hide information while machine learning looks to discover information, yet the fields overlap in many applications [1]. Medical data in particular requires a high level of privacy and HIPAA constraints were designed in order to provide patients with a necessary level of confidentiality. On the other hand, these records represent a wealth of data that has already been collected by various research institutions and hospitals.

Delaram Kahrobaei and Vladimir Shpilrain are partially supported by ONR (Office of Naval Research) grant N00014-15-1-2164.

Alexander Wood is with the Department of Computer Science, The Graduate Center, CUNY, New York, NY, 10016, USA; The Department of Computational Medicine and Bioinformatics, University of Michigan Center for Integrative Research in Critical Care (MCIRCC), Ann Arbor, MI, 48109, USA; and the Emergency Medicine Department, University of Michigan, Ann Arbor, MI, 48109, USA (e-mail: awood@gradcenter.cuny.edu)

Vladimir Shpilrain is with the Department of Mathematics, The Graduate Center, CUNY, New York, NY, 10016, USA, and the Department of Mathematics, The City College of New York, New York, NY, 10031, USA (email: shpil@groups.sci.ccny.cuny.edu)

Kayvan Najarian is with the Department of Computational Medicine and Bioinformatics, University of Michigan Center for Integrative Research in Critical Care (MCIRCC), Ann Arbor, MI, 48109, USA, and Emergency Medicine Department, University of Michigan, Ann Arbor, MI, 48109, USA (e-mail: kayvan@med.umich.edu)

Delaram Kahrobaei is with the Department of Computer Science, Tandon School of Engineering, New York University, New York, NY, USA, and the Department of Computer Science, The Graduate Center, CUNY, New York, NY, 10016, USA (e-mail: dk2572@nyu.edu)

The ability to use this information without compromising the patients' privacy would streamline collaboration between researchers and clinicians. Privacy concerns make it difficult for researchers to access large amounts of medical data. However, training classification models on a small or single-source data set can lead to overfitting or over generalization of the model's accuracy during the testing stage. This results in a model which is not generalizable to new data [2].

The ability to perform private classification, where researchers test a trained classification model on an encrypted external database without revealing the model, would provide medical researchers with a method of verifying the generalizability of their models. Another valuable application for private classification is for assisted decision making and diagnosis systems. The owners of these systems do not wish to share their models, and clinicians do not wish to share their patients' data points. However, access to a well-trained model could assist clinicians in diagnosing disease in their patients.

Fully-homomorphic encryption (FHE), which allows for computation of arbitrary functions over encrypted data, is one approach towards enabling private classification algorithms. Current FHE research focuses on public-key cryptosystems which involve public encryption key and a private decryption key. Some of these applications focus specifically on genomic computation, including edit distance [3], string matching [4], [5], genomic tests such as ancestry and paternity [6], and other genomic tests [7], [8]. Other research has focused on the task of private classification, including neural networks [9], [10], decision trees [11], and Fisher's linear discriminant classifier [1].

In contrast, private key cryptosystems require prior knowledge of the encryption/decryption key(s). In other words, if multiple parties wish to perform decryption in a private-key setting, they must first exchange keys over a secure channel. While this is considered a disadvantage when the goal is purely communication, it is not necessarily a disadvantage within medical applications due to different privacy concerns [12]. In this paper we propose a fully homomorphic private key protocol for private Naive Bayes classification, a private argmax protocol, and test the protocol on publicly available breast cancer data [13]. We classify each data point with an average time of approximately half of a second.

A. Fully Homomorphic Encryption

Fully homomorphic encryption was first conceptualized as a 'privacy transformation' which would allow for computing functions on encrypted data without first having to decrypt the information [14]. This concept is known today as a fully homomorphic encryption scheme, formally defined as an encryption scheme which allows for computation of arbitrary functions over encrypted data. Because a boolean circuit can be used to describe arbitrary computations, a scheme only needs to be homomorphic over addition and over multiplication in order to be described as fully homomorphic [15]. A scheme is called additively homomorphic if

$$[\![x + y]\!] = [\![x]\!] \oplus [\![y]\!]$$

and multiplicatively homomorphic if

$$[\![x \cdot y]\!] = [\![x]\!] \otimes [\![y]\!]$$

for operations $\oplus, \otimes$ in the ciphertext space, where $[\![\alpha]\!]$ denotes the encryption of α. While computation of arbitrary functions is *theoretically* possible using addition and multiplication as described above, this in itself is not sufficient for *practical* computation of arbitrary functions. Any fully homomorphic encryption scheme which is suited for practical use will only be able to compute polynomial functions and polynomial approximations of functions. This has been termed *polynomial machine learning* in the context of fully homomorphic encryption [1].

Gentry's first fully homomorphic encryption scheme is lattice-based [16]. Improvements on this method led quickly to the second generation of fully homomorphic encryption schemes [17], [18], [19] including the BGV scheme [20], [17], [18] and YASHE [21]. Recent improvements have built upon this foundation to yield even faster schemes which allow for practical applications [22], [23], [24], [25], [26], [27].

B. GKS FHE Scheme

The GKS scheme is a ring-based private-key FHE scheme is able to avoid much of the computational overhead required for fully homomorphic public key encryption and is secure against ciphertext-only attack [12]. Define the ring S_n as

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

This ring contains a super-exponential (in n) number of *idempotent* elements, or elements $g \in S_n$ such that $g^2 = g$. Some idempotent elements are given by

$$e_F := \prod_{i \in F} x_i \cdot \prod_{j \notin F} (1 - x_j)$$

for any set of indexes $F \subseteq \{1, 2, \dots, n\}$. The above construction yields 2^n idempotents which are pairwise orthogonal, meaning $e_F e_G = 0$ whenever $F \neq G$, and represent a linear basis of S_n over $\mathbb{Z}_p$.

The scheme is ring-based and involves a private plaintext ring P and a ciphertext ring C, where P is a retract of C, i.e., $P \subset C$ and at the same time P is a factor ring of C. Both P and C are rings of the above form. Let I be an ideal of C such that C/I is isomorphic to P.

The data owner, D, generates C and I during the key generation algorithm within the scheme. Parameters n, r, and p are chosen such that p is a large prime, $r > n$, and let $P := S_n$. The ciphertext ring $C = S_r$ is randomly generated from $P = S_n$ by $\{x_i\}_{i=n+1}^{r}$, where $r > n$. In other words, the ring $C = S_r$ expands upon the generators of $P = S_n$. The data owner then chooses an ideal I by randomly selecting elements

$$x_m - w_m(x_1, x_2, \dots, x_{m-1})$$

for $m = n+1, \dots, r$, where $w_m(x_1, x_2, \dots, x_{m-1})$ represents an idempotent element of S_{m-1}. Observe that C/I is isomorphic to P by construction. The data owner next rewrites C in terms of its orthogonal basis $\{e_i\}_{i=1}^{2^r}$ and applies a random permutation π to the orthogonal set generators. D concludes the key generation algorithm with a final transformation between this permuted basis, given by

$$C = \langle e_1, e_2, \dots, e_{2^r} | p \cdot 1 = 0, e_i^2 = e_i, \\ \text{and } e_i e_j = 0 \text{ for all } i, j \rangle,$$

and a *triangular basis*. This triangular basis is initially constructed by letting

$$t_m = \sum_{i=1}^{k} e_k$$

for $1 \leq k \leq 2^r$. Next, m rows are randomly repeated in this basis and the final, public, ciphertext ring is given by

$$\tilde{C} = \langle t_1, t_2, \dots, t_{2^r+m} | p \cdot 1 = 0, e_i^2 = e_i, \\ \text{and } e_i e_j = 0 \text{ for all } i, j \rangle.$$

Note that in the published basis, homomorphic addition occurs component- wise and the homomorphic multiplication operation is given by $t_i t_j = t_i$ whenever $i \leq j$.

Elements in the orthogonal basis are not uniquely represented by elements in the triangular basis. For example, if $t_1 = e_1$, $t_2 = e_1$, $t_3 = e_1 + e_2$, and $t_4 = e_1 + e_2$, then the element e_2 in the orthogonal basis can be written in the triangular basis in four ways: $t_3 - t_1$, $t_3 - t_2$, $t_4 - t_1$, or $t_4 - t_2$. This is used to the key holder's advantage during encryption.

A plaintext element $x \in P$ is encrypted as

$$[\![x]\!] = x + E_0$$

where

$$E_0 = \sum_{j=n+1}^{r} (x_j - w_j(x_1, \dots, x_{j-1})) \cdot h_j(x_1, \dots, x_r)$$

for random $h_j \in C$. The element E_0 therefore belongs to the ideal I. Covert $[\![x]\!]$ from the standard basis to the orthogonal basis, then to the triangular basis. Because the orthogonal elements do not necessarily have a unique expression in the triangular basis, the triangular expression of each orthogonal basis element is randomly selected from possible representations during encryption.

To decrypt, the data owner simply converts the ciphertext back to the basis $\{x_i\}_{i=1}^{r}$, and then replaces x_j with $w_j(x_1, \dots, x_{j-1})$ for $j = n+1, \dots, r$.

II. Privacy-Preserving Classification

The term *privacy-preserving classification* (PPC) was coined in [11] and broadly describes a situation in which a user is able to classify her datapoint using a data owner's learned model, while neither party learns any information about the other party's data. A major application of PPC lies in the medical field. With PPC, a patient can perform medical analyses of her medical data without revealing her personal information.

Differential privacy [28] is one approach to which has seen some success with training various classification algorithms. This method, however, is built upon having a large database and loses its utility when the goal is to classify a *single* datapoint [9]. Leveled homomorphic encryption schemes (LHE), a variant on fully homomorphic encryption which allow for polynomial computation up to a predefined depth, such as YASHE have been used in an application of neural networks to encrypted data called CryptoNets [9]. ML Confidential [1] used LHE to run classification using Linear Means and Fisher's Linear Discriminant classifiers.

In addition, there are methods which are not based entirely on fully homomorphic encryption. The authors in [11] construct efficient protocols for privacy-preserving classification via hyperplane detection, Naive Bayes, and decision trees using two additively homomorphic encryption schemes, Quadratic Reciprocity (QR) [29] and Paillier [30], and one leveled homomorphic encryption scheme, HELib [27].

A. Private Naive Bayes

The Naive Bayes classifier is based off of Bayes Theorem, which states that

$$\Pr(G=\ell|X)=\frac{\Pr(X|G=\ell)\Pr(G=\ell)}{\Pr(X)}$$

where C denotes the class of a data point X. In other words, we compute the posterior probability $\Pr(C=\ell|X)$ using prior knowledge combined with observed data.

The Naive Bayes model assumes that the inputs are conditionally independent in each class, hence the "naive" title. Naive Bayes is able to provide fast results which surprisingly are often better than more sophisticated methods. The naive Bayes classifier does not consider correlation among features, thus lowering the variance while increasing the bias and hence avoiding overfitting on the training set. Due to this it has become a benchmark used by the community in data analysis.

The classification of a vector X is given simply by taking the maximum posterior probability over all classes and applying Bayes' Theorem:

$$\operatorname*{argmax}_{G_i\in G}\Pr(G=G_i)\prod_{k=1}^{p}\Pr(X=X_k|G=G_i).$$

Commonly in medical applications each attribute X_i of a vector X can take on only a discrete set of values. Then, all of the information needed to classify an arbitrary data point can be succinctly represented in a collection of vectors and matrices. This representation will enable us to collect all data needed in order to perform a classification. The ease with which all this data can be represented will be of great use in the protocol which follows. Let $g=(g_j)$ be a vector where

$$g_j=\Pr(G=G_j)$$

and let $h=(h_{i,j})$ be a matrix where

$$h_{i,j}=\Pr(X=X_i|G=G_j).$$

Given tables g and h, any new data point could be classified by looking up the probability for each attribute, given each class, and computing the `argmax` of the resulting values.

Bost et. al. implement private naive bayes classification between a client and a service provider [11]. The client has a single data vector x which she would like to keep private. The service provider owns a private model w. Given a classification algorithm C the client should be able to learn $C(x,w)$, the classification of her vector x using the model w, without learning any partial information about the model or giving away any information about her input.

Let pk_Q, sk_Q denote a public and secret key pair in the QR scheme and pk_P, sk_P denote the same in Pallier. Let square brackets $[\![a]\!]$ denote the encryption of an value a. The security of these schemes provides semantic security for the authors' protocols, and an honest-but-curious adversary model is used.

Let $\mathcal{G}$ denote the set of classes and assume there is a finite number of values each attribute can take. Let P denote the vector of class probabilities, where $P_i=\Pr(G_i)$, and let T denote the collection of tables given by $T_{i,j}(X)=\Pr(X=X_i|G=G_i)$. Assume each vector X has d features, and that there are c possible classes. The authors' protocol runs as follows:

```
1: The service provider encrypts the tables $P$ and $T$ using Pallier.
2: The server sends the encrypted tables to the client.
3: The client computes $[\![p_i]\!]=[\![P_i]\!]\prod_{j=1}^{d}[\![T_{i,j}(x_j)]\!]$ for $i=1,\ldots,c$.
4: The client uses the server to compute $i=\mathrm{argmax}_i p_i$.
5: Client outputs $i$.
```

Note that Step 3 of the above protocol requires only that the encryption scheme be multiplicatively homomorphic. It is Step 4 which would ultimately require an additively homomorphic scheme. The authors in [11] use multiple encryption methods, combined with an algorithm for changing the encryption scheme, in order to compute the argmax. In the next section, we describe our adapted version of the protocol which performs classification using a single encryption scheme.

III. Proposed Method for Fully Homomorphic Private Classification With Naive Bayes

The model described below varies in several slight but important ways from the previous protocol. First, as stated previously, this protocol is designed for an encryption scheme which is both private-key and fully homomorphic. Furthermore, our model assumes direct communication between the service provider and the data owner. In other words, there is not a trusted server acting as intermediary between the parties. The protocol could easily be adapted to include a cloud service provider to carry out computations, if desired.

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

There is a certain amount of information which Alice *must* know in order to carry out the protocol. Namely, Alice must know that the data vector X has p features and that there are exactly r classes. However, she should not be able to deduce any information about the conditional class probabilities associated with q features, or the r class probabilities.

Because Bob has already computed a learned model, he prepares two tables. First, Bob prepares table P represented as a column vector of degree r where $P_i = \Pr(G = G_i)$, the prior probability on class G_i. Next he prepares a table T as a $r \times q$ matrix where entry T_{ij} represents $\Pr(X = X_j | G = G_i)$.

Private fully homomorphic Naive Bayes classification proceeds as follows:

1: Bob prepares the tables P and T and publishes their encryptions, $[\![P]\!]$ and $[\![T]\!]$.

2: For each class G_i for i from 1 to r, Alice computes

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

3: Alice computes

$$i = \operatorname*{argmax}_{1 \le i \le r} [\![\Pr(G_i|X)]\!]$$

using the private argmax protocol described below in Protocol 2 and outputs i.

See Figure 1 for an illustration of this protocol. There are two parties to consider when discussing the security of this protocol: the privacy of the Data Owner Alice's information as well as the privacy of the Classification Model Owner Bob's learned model. The privacy of the learned model is derived entirely from the security of the encryption scheme used. Discussion of the privacy of Alice's information, however, requires knowledge of the argmax protocol called in Step 3.

Previous approaches to private argmax using public-key encryption allow Alice to mask her value with random noise encrypted under Bob's public key. This approach will not work with private-key encryption because Alice cannot encrypt random noise.

We describe our algorithm for computing argmax in a private-key FHE protocol via a series of improvements on an initial protocol which leaks information. Denote $\mathcal{P}_i = \Pr(G_i|X)$, $\mathcal{E}_i = [\![\mathcal{P}_i]\!]$, and the set of all $\mathcal{E}_i$ as $\mathcal{E}$. First, suppose Alice computes argmax by sending the set of encrypted probability values $\mathcal{E}$ to M. Then, Bob decrypts each value and sends Alice the index of the highest value using an asymmetric encryption protocol to hide the value of the index from an eavesdropper. While Alice has learned nothing about the encrypted model, Bob learns not only which class Alice's datapoint belongs to but also the exact probabilities for each class.

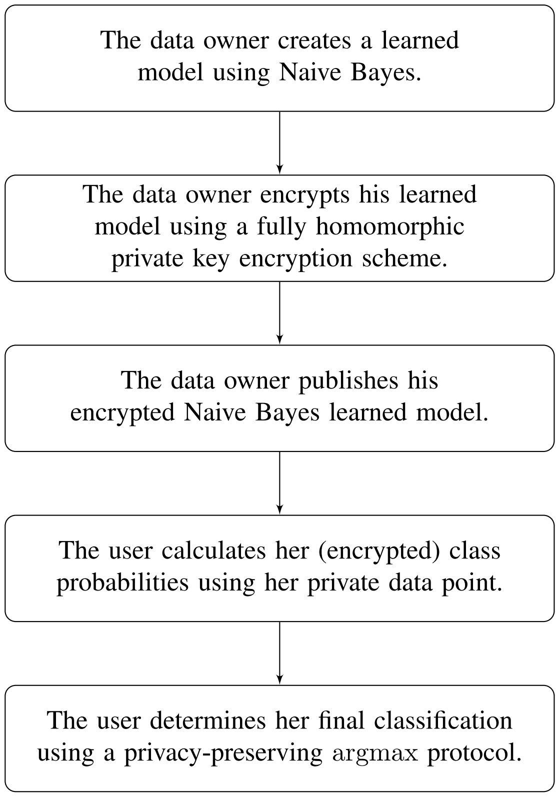
The data owner creates a learned model using Naive Bayes.
The data owner encrypts his learned model using a fully homomorphic private key encryption scheme.
The data owner publishes his encrypted Naive Bayes learned model.
The user calculates her (encrypted) class probabilities using her private data point.
The user determines her final classification using a privacy-preserving argmax protocol.

Fig. 1: Private Naive Bayes Classification Outline

Suppose instead that Alice performs a permutation π on the class probabilities then sends $\pi(\mathcal{E})$ to Bob. Bob decrypts the values, determines which is largest, and sends that index to Alice, who reverses the permutation to determine her class. Note that it is not necessary to hide the value of the index sent to Alice from any eavesdroppers in this scenario, as the permutation π randomized the value. However, while this method prevents Bob from determining which probability is associated with each class specifically it does not prevent him from learning the class probabilities themselves.

Next, Alice attempts to hide the class probabilities by breaking down the computation of argmax to a series of comparisons. In particular, she randomly selects two encrypted probabilities $\mathcal{E}_i$, $\mathcal{E}_j \in \mathcal{E}$. She then sends the value

$$\mathcal{E}^* = \mathcal{E}_i - \mathcal{E}_j = [\![\mathcal{P}_i - \mathcal{P}_j]\!]$$

to Bob. Now when Bob decrypts $\mathcal{E}^*$ he learns the value $\mathcal{P}_i - \mathcal{P}_j \in (-p/2, p/2]$, but not either of the probabilities themselves. If this value is negative, Bob sends the bit $b = 0$ to Alice, and sends $b = 1$ otherwise. Bob is then able to determine which value was the larger one based on the initial selection of $\mathcal{P}_i$ and $\mathcal{P}_j$ and rules out the smaller of the two values. The procedure is repeated until only one value remains.

Because the value sent to Bob is randomly ordered, publishing the bit 0 or 1 will also appear random to any observer hence does not compromise privacy. However, Bob recovers some partial information about Alice's data. He recovers a collection of r values representing the difference between pairs of the posterior probabilities for different $i \in [r]$. While permuting the elements keeps him from knowing which pairs of elements correspond to which values, it is not outside the realm of possibility that he could learn partial information about Alice's private data using this information.

Therefore, Alice needs to mask the value given by the difference between each pair of probabilities. To execute this, we take advantage of the fully homomorphic properties of our protocol by constructing a family $\mathcal{F}$ of additively homomorphic, *monotone functions* that commute with encryption. A function $f : R \to R$ *commutes* with encryption if

$$f\left(\llbracket m \rrbracket\right) = \llbracket f(m) \rrbracket.$$

The additively homomorphic property of f guarantees that $f(m+n) = f(m) + f(n)$.

A privacy-preserving algorithm for argmax using a family of additively homomorphic monotone functions, $\mathcal{F}$, proceeds as follows:

```
1: Set $I = \{1, 2, \ldots, r\}$.
2: **while** $|I| > 1$ **do**
3:     $A$ computes a random permutation $\pi$ on $I$.
4:     $A$ randomly chooses $f \leftarrow \mathcal{F}$ and computes the values $f\left(\mathcal{E}_{\pi(1)}\right)$ and $f\left(\mathcal{E}_{\pi(2)}\right)$.
5:     $A$ uses the additive homomorphic properties of encryption and $f$ as well as the commutative property of the function to evaluate
           $\mathcal{E}^* = f\left(\mathcal{E}_{\pi(1)}\right) - f\left(\mathcal{E}_{\pi(2)}\right)$
               $= f\left(\llbracket \mathcal{P}_{\pi(1)} \rrbracket\right) - f\left(\llbracket \mathcal{P}_{\pi(2)} \rrbracket\right)$
               $= \llbracket f(\mathcal{P}_{\pi(1)} - \mathcal{P}_{\pi(2)}) \rrbracket$
6:     $A$ sends $\mathcal{E}^*$ to $B$.
7:     $B$ decrypts $\mathcal{E}^*$ and recovers
           $f(\mathcal{P}_{\pi(1)} - \mathcal{P}_{\pi(2)})$
       If this value is negative, $B$ sends the bit $b = 0$ to $A$, otherwise send $b = 1$.
8:     If $b = 0$, remove $\pi(1)$ from $I$. Otherwise remove $\pi(2)$.
9: **end while**
10: $A$ returns $I$.
```

During this protocol, B (Bob) 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 him from learning partial information from the decrypted value.

IV. Implementation

We run the above protocol using the GKS encryption scheme, where coefficients in $\mathbb{Z}_p$ are taken from $(-p/2, p/2]$ and correspond to positive and negative integers, and

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

for a randomly chosen integer k up to 20 bits. Our implementation uses a 198-bit prime p.

We implemented the protocols in C++ on a MacBook Pro using El Capitan, a 2.3 GHz Intel Core i7, with 16 GB memory. We used GNU Multiple Precision Library (GMP) [31] to allow for integer storage above the built-in data type limits in C++.

A. Encoding

Fully homomorphic computation can only be utilized over a fully homomorphic encoding. The authors of [12] provide a straightforward method of implementing a fully homomorphic encoding. Any fully homomorphic embedding of $\mathbb{Z}_p$ into the ring S_n will suffice. Specifically, it is sufficient to map the element 1 of $\mathbb{Z}_p$ to any idempotent element of S.

Elements in the scheme have coefficients which lie in the integer field $\mathbb{Z}_p$. Therefore floating point numbers must be encoded as integers in $\mathbb{Z}_p$. 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. We can scale a floating point value to yield n levels of precision via an encoding function $\texttt{Encode}(x) := \lfloor x \cdot 10^n \rfloor$. Then, this encoding is extended to the ring S_n by mapping x to a fixed idempotent value in S and assigning random values in $\mathbb{Z}_p$ to the remaining coefficients.

To decode, retrieve the coefficient of the fixed idempotent value in S and multiply by 10^{-n}. This encoding requires the user to keep track of the "depth" of each element. The depth of an encoded value is initialized to $d = 1$, and each time we perform multiplication its depth is incremented. To decode, multiply the coefficient by 10^{-dn}. Only elements which share a depth may be multiplied. To increase the depth of an element, simply multiply it by the scalar 10^n.

This encoding suffers from overflow over the prime modulus p if too many units of decimal precision are required. With each multiplication performed, the size of the input grows exponentially. Encoded numbers with depth d can take on values up to 10^{dn}.

The model in our experiments was trained with five decimal points precision, and the unencrypted experimental results were obtained with five decimal points precision. In the encrypted experiments all values were initially encrypted with three decimal points precision. Table I shows that no loss in accuracy resulted from this change in decimal precision.

V. Results on Breast Cancer Classification

Data from the UCI Machine Learning Repository was used to test the performance of the protocols [13]. 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. These nine attributes include clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. A clinician measured and rated each

TABLE I: Unencrypted Versus Encrypted Experimental Results under 10×10-fold Cross Validation

	Time (s)	Accuracy	Sensitivity	Specificity	Precision	NPV	F1-score
Unencrypted	0.00001	0.96003	0.93389	0.97410	0.95100	0.96476	0.94292
Encrypted	0.49215	0.96003	0.93389	0.97410	0.95100	0.96476	0.94292

attribute on a scale of 1 to 10 at the time it was collected. Lower values correspond to what clinicians expect to see in a benign case and higher values correspond to the malignant case. Previous research has 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 [32].

We implemented a Naive Bayes algorithm to create a learned model and encrypted this learned model using the GKS Encryption Scheme. We performed 10×10-fold cross validation to evaluate the performance of the learned model. Furthermore, because the data set is unbalanced with a higher proportion of benign tumors, we performed oversampling while training the models. A positive case denotes a case where the tumor is diagnosed as malignant and a negative case refers to a tumor which is benign.

We performed additive smoothing on the values in the tables of class and prior probabilities prior to testing to prevent error in the case of zero or near zero probabilities. Specifically, each probability was increased by 0.1, and any value which was greater than or equal to 1 after smoothing was reset to 0.999. The size of the ciphertext ring in the experiments was $2^8 + 50 = 306$, and the prime modulus p was 198 bits.

We tested classification data points not included in the training set in both encrypted and unencrypted formats and performed 10×10-fold cross validation. Table I shows a comparison the average performance of the encrypted and unencrypted experiments. Results include classification time for a single data point in seconds, accuracy, sensitivity, precision, specificity, and the true negative rate.

VI. Discussion

The time increase between encrypted and unencrypted computation is expected and occurs in all current fully homomorphic encryption methods. For the example provided above, where a single user wishes to classify their data, classification in approximately half a second is well within a reasonable time range for medical applications. The minimum classification time of an encrypted data point observed was 0.45146 seconds, and the maximum time was 0.59803 seconds. The authors in [11] report a similar classification time of 0.479 seconds on the same data set, although it is not clear how large of a testing set was used to generate this score. They do not provide other information on the classifier's performance.

Other statistics we provide include Sensitivity, Precision, Specificity, and the True Negative Rate (TNR). In the case of breast cancer specifically it is crucial to positively identify all malignant tumors for timely medical intervention. Our results yield high precision and show higher sensitivity, meaning there is a low rate of false positives and an even lower rate of false negatives.

Specificity and the True Negative Rate (TNR) are also known as Inverse Recall and Inverse Precision, as they provide similar information for negative classifications. Specificity describes the proportion of negative cases which were classified as negative and TNR describes the proportion of cases classified as negative which were negative in reality. Our results show both high Specificity and high TNR. The higher of the two is TNR, which is ideal for medical applications as a false negative could have dire consequences for the patient.

VII. Conclusions

Private-key fully homomorphic encryption can classify medical data efficiently. Similar techniques could enable researchers and clinicians to utilize private medical data and models which they cannot access in the clear. Hospitals and companies with trained assisted diagnosis systems could use this technology to provide access to their models without giving away their parameters, and doctors can use this software without revealing their patient's information. Medical researchers can use these methods to determine whether their models are over fit to their own data sets,

References

[1] T. Graepel, K. Lauter, and M. Naehrig, *ML Confidential: Machine Learning on Encrypted Data*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–21.

[2] H. N. A. Pham and E. Triantaphyllou, *The Impact of Overfitting and Overgeneralization on the Classification Accuracy in Data Mining*. Boston, MA: Springer US, 2008, pp. 391–431.

[3] J. H. Cheon, M. Kim, and K. Lauter, *Homomorphic Computation of Edit Distance*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 194–212.

[4] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik, "Privacy preserving error resilient dna searching through oblivious automata," in *ACM Conference on Computer and Communications Security (CCS)*, ACM Press. Alexandria, Virginia, USA: ACM Press, Oct 29–Nov 2 2007, p. 519–528.

[5] E. Ayday, J. L. Raisaro, U. Hengartner, A. Molyneaux, and J.-P. Hubaux, *Privacy-Preserving Processing of Raw Genomic Data*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 133–147.

[6] F. Bruekers, S. Katzenbeisser, K. Kursawe, and P. Tuyls, "Privacy-preserving matching of dna profiles," Tech. Rep., 2008.

[7] K. Lauter, A. López-Alt, and M. Naehrig, *Private Computation on Encrypted Genomic Data*. Cham: Springer International Publishing, 2015, pp. 3–27.

[8] M. Kim and K. Lauter, "Private genome analysis through homomorphic encryption," Cryptology ePrint Archive, Report 2015/965, 2015, http://eprint.iacr.org/2015/965.

[9] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy," in *PMLR*, Jun. 2016, pp. 201–210.

[10] R. Shokri and V. Shmatikov, "Privacy-Preserving Deep Learning," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: ACM, 2015, pp. 1310–1321.

[11] R. Bost, R. Ada Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data." Symposium on Network and Distributed System Security (NDSS), February 2015.

[12] A. Gribov, D. Kahrobaei, and V. Shpilrain, “Private-key fully homomorphic encryption in rings,” *Groups, Complexity, Cryptology*, vol. 10, 2018.

[13] M. Lichman, “UCI machine learning repository,” 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[14] R. Rivest, L. Adleman, and M. Dertouzos, “On Data Banks And Privacy Homomorphisms,” *Foundations of Secure Computation*, vol. 4, no. 11, pp. 165–179, 1978.

[15] C. Peikert, “A Decade of Lattice Cryptography,” *Foundations and Trends in Theoretical Computer Science*, vol. 10, no. 4, pp. 283–424, Mar. 2016.

[16] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, ser. STOC ’09. ACM, 2009, pp. 169–178.

[17] Z. Brakerski and V. Vaikuntanathan, “Fully Homomorphic Encryption from ring-LWE and Security for Key Dependent Messages,” in *Proceedings of the 31st Annual Conference on Advances in Cryptology*, ser. CRYPTO’11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 505–524.

[18] ——, “Efficient Fully Homomorphic Encryption from (Standard) LWE,” in *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, ser. FOCS ’11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 97–106.

[19] M. v. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, “Fully Homomorphic Encryption over the Integers,” in *Advances in Cryptology – EUROCRYPT 2010*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, May 2010, pp. 24–43.

[20] Z. Brakerski, V. Vaikuntanathan, and C. Gentry, “Fully homomorphic encryption without bootstrapping,” in *In Innovations in Theoretical Computer Science*, 2012.

[21] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig, *Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 45–64.

[22] J. Alperin-Sheriff and C. Peikert, “Faster Bootstrapping with Polynomial Error,” in *Advances in Cryptology – CRYPTO 2014*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Aug. 2014, pp. 297–314.

[23] C. Gentry, A. Sahai, and B. Waters, “Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based,” in *CRYPTO*. Springer, 2013, pp. 75–92.

[24] C. Gentry, S. Halevi, C. Peikert, and N. P. Smart, “Ring Switching in BGV-Style Homomorphic Encryption,” in *Security and Cryptography for Networks*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Sep. 2012, pp. 19–37.

[25] C. Gentry, S. Halevi, and N. P. Smart, “Fully Homomorphic Encryption with Polylog Overhead,” in *Advances in Cryptology – EUROCRYPT 2012*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Apr. 2012, pp. 465–482.

[26] ——, *Better Bootstrapping in Fully Homomorphic Encryption*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–16.

[27] S. Halevi, “HElib: An Implementation of homomorphic encryption,” 2013. [Online]. Available: https://github.com/shaih/HElib

[28] C. Dwork, *Differential Privacy*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12.

[29] S. Goldwasser and S. Micali, “Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information,” in *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, ser. STOC ’82. New York, NY, USA: ACM, 1982, pp. 365–377.

[30] P. Paillier, “Public-Key Cryptosystems Based on Composite Degree Residuosity Classes,” in *Advances in Cryptology — EUROCRYPT ’99*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, May 1999, pp. 223–238.

[31] “The GNU MP Bignum Library.” [Online]. Available: https://gmplib.org/

[32] W. H. Wolberg and O. L. Mangasarian, “Multisurface method of pattern separation for medical diagnosis applied to breast cytology,” *Proceedings of the National Academy of Sciences, U.S.A.*, vol. 87, pp. 9193–9196, 1990.