PROC. INTERNAT. CONF. SCI. ENGIN. ISSN 2597-5250
Volume 3, April 2020 | Pages: 397-403 E-ISSN 2598-232X

Secure Naïve Bayes Classification without Loss of Accuracy with Application to Breast Cancer Prediction

Artrim Kjamilji[1], Arben Idrizi[2], Shkurte Luma-Osmani[3], Ferihane Zenuni-Kjamilji[4]
[1]Faculty of Engineeringand Natural Sciences, Computer Science and Engineering, Sabancı University,Istanbul-Turkey
[2]Faculty of Natural Sciences and Mathematics, Graduate school of Mathematics, University of Tetovo, Tetovo-North Macedonia
[3]Faculty of Natural Sciences and Mathematics, Bureau of Innovations, University of Tetovo, Tetovo-North Macedonia
[4]Faculty of Philosophy, Professional Education, University of Tetovo, Tetovo-North Macedonia
E-mails: artrimk@sabanciuniv.edu[1], aidrizi@hotmail.com[2] shkurte.luma@unite.edu.mk[3], ferihane.zenuniqamili@gmail.com[4]

Abstract. The classification and prediction accuracy of Machine Learning (ML) algorithms, which often outperform human experts of the related field, have enabled them to be used in areas such as health and disease prediction, image and speech recognition, cyber-security threats and credit-card fraud detection and others. However, laws, ethics and privacy concerns prevent ML algorithms to be used in many real-case scenarios. In order to overcome this problem, we introduce a few flexible and secure building blocks which can be used to build different privacy preserving classifications schemes based on already trained ML models. Then, as a use-case scenario, we utilize and practically use those blocks to enable a privacy preserving Naïve Bayes classifier in the semi-honest model with application to breast cancer detection. Our theoretical analysis and experimental results show that the proposed scheme in many aspects is more efficient in terms of computation and communication cost, as well as in terms of security properties than several state of the art schemes. Furthermore, our privacy preserving scheme shows no loss of accuracy compared to the plain classifier.

Keywords: privacy-preserving, Naive Bayes, classification, breast cancer prediction

INTRODUCTION

Many modern technologies, such as cloud computing, wearable and ubiquitous computing, Internet of Things (IoT) and others, have enabled us to collect huge amount of data and come to what is now known as Big Data, were annually it is expected to be generated 44 zettabytes of data only throughout 2020 alone [1]. This data hides in itself important patterns and information that the human brain cannot comprehend. Nevertheless, rapid advancements of the computational power of modern processors and computer systems, combined with the increasing network speeds, have enabled us to take benefit of those deeply hidden patterns and throve of information. Especially we see this benefit when we use this data for the purpose of training (building) Machine Learning (ML) models (algorithms), which in their prediction often surpass human experts of the corresponding field [2]. Those algorithms include deep learning (neural networks), support vector machines (SVM), Naïve Bayes, decision trees, random forests, etc. However, there is a drawback. Due to law and privacy requirements, no entity would be comfortable to publicly share their data for the purpose of training ML models. Rather, they would like to do it in what is called as **privacy preserving training** where, roughly, those entities use privacy-preserving techniques to train ML models which enables them to hide their data during the training process. On the other hand, after we have obtained the trained model, we face the same privacy concerns during the prediction phase. Namely, the user (client) that has unclassified data doesn't want to reveal this data neither the final prediction (classification) to the server that holds the trained model, while the server doesn't want to reveal any parameter of the trained model to the user. This process is called **privacy preserving classification (prediction)**.

Nearly a couple of decades ago, [3-4] almost at the same time came with the notion of privacy preserving. Throughout the years many schemes have been proposed that deal with either privacy preserving training or classification or both for various machine learning algorithms. Some of them include privacy preserving deep learning [2,5], decision trees [6-8], hyperplane systems and SVM [6-8], Naïve Bayes [6-10] and others.
In this paper we deal only with privacy preserving Naïve Bayes classifiers and leave for the follow-up papers the rest of ML algorithms. Due to space constraints here we will exclusively address the issue of privacy preserving classification based on the Naïve Bayes algorithm and leave open the issue of privacy-preserving training with multiple dataset owners involved for an extension of this paper.

PRELEMINARIES AND PARTICIPANTS

Naïve Bayes classifier

Let's say that each of the dataset's records (transactions, rows, instances) have f features. Each of those feature sets can have certain values, i.e. $Fi = \{V_{1,Fi}, V_{2,Fi}, \ldots, V_{|Fi|,Fi}\}$, where |Fi| is the cardinality (number of elements) of

set Fi and $v_{j,Fi}$ is it's j-th element of set Fi, with $1 \le i \le f$ and $1 \le j \le |Fi|$. Hence each of the dataset record belongs to theset of features $F \subseteq R^f$ which is a Cartesian product of the features, hence $F=\{F_1 x F_2 x \ldots x F_f\}$.All of the instances are labeled (belong to) one of the c classes from the set of classes $C=\{C_1, C_2, \ldots C_c\}$. In total we have NT transactions (records) in the dataset. We denote with $N(C_k)$the frequency (number) of transactions that belong to class C_k and with $N(V_{j,Fi}; C_k)$ the frequency of transactions that has label C_k for the j-th value of the feature Fi. Correspondingly, we denote the class probabilities as $P(C_k) = \frac{N(C_k)}{NT}$and the conditional value-class probabilities as $P(V_{j,Fi}|C_k) = \frac{N(V_{j,Fi};C_k)}{N(C_k)}$, where $1 \le i \le f$, $1 \le j \le |Fi|$ and $1 \le k \le c$. Those probabilities actually represent the Naïve Bayes trained model.

For an unclassified (un-labeled) query vector $X=\{X_1, X_2, \ldots X_f\}$, where $X \in F$ (hence $X_i \in F_i$, for $1 \le i \le f$), we denote as C(X) the process of assigning a label (classifying) this query according to the maximum likelihood $C(X) = argmax_{1 \le j \le c} P(Cj|X)$. Using the Naïve Bayes formula:

$$C(X) = argmax_{1 \le k \le c} \frac{P(C_k) P(x_1, \ldots x_f | C_k)}{P(x_1, \ldots x_f)} \quad (1)$$

since the term $P(x_1, \ldots x_f)$ is the same for all classes, then naively assuming that all of the features are independent between each-other, hence $P(x_1, \ldots x_f | C_k) = \prod_{i=1}^{f} P(X_i|C_k)$, then (1) can be re-written as:

$$C(X) = argmax_{1 \le k \le c} P(C_k) \prod_{i=1}^{f} P(X_i|C_k) \quad (2)$$

Despite its' naïve assumptions, yet Naïve Bayes is among most widely used ML classifiers due to its' simplicity and high prediction accuracy [9-10].

Public Somewhat homomorphic encryption

Public encryption schemes allow asymmetric encryption techniques were there is a pair of two keys for encryption/decryption, one is public and the other one is kept secret. If a certain message is encrypted with one key, it can be decrypted with the other one. Public somewhat homomorphic encryption (SWHE) schemes allow certain operations (such as additions and multiplications) to be done on the ciphertexts without decrypting them, which are known as homomorphic operations.Most SWHE schemes work with integers. In this sense, we denote by [.] an encryption of an integer message using a SWHE scheme, hence [ctxt]=Encrypt(m), where Encrypt(.) is the encryption function and m is a plaintext message. Hence for the homomorphic operations we have $[ctxt_3]=[ctxt_1]+[ctxt_2]$, and $[ctxt_4]=[ctxt_1]x[ctxt_2]$. SWHE schemes also allow for operations between plaintexts and ciphertexts, i.e. $[ctxt_5]=ptxt+[ctxt_1]$ and $[ctxt_6]=ptxt+[ctxt_1]$. The number of multiplications though is limited and it's known as the circuit depth of the scheme. The decryption function is denoted Decrypt(.), thus m = Decrypt([ctxt]). SWHE schemes are mostly based on hard problems on lattices, such as Ring Learning with Error (RLWE) [11]. Using the Chinese Remainder Theorem (CRT), Smart et.al [12] enabled a Single Instruction Multiple Data (SIMD) fashion of homomorphically executing the operations over encrypted data (fig.1). This opened the way for huge computation improvements by executing the homomorphic operations in parallel in component (slot-wise) manner, with no extra cost(fig.1). This is especially helpful for ML algorithms, which often do several operations that are similar (of same nature) with each-other, so they can be done in SIMD fashion. Integers are encoded on polynomial rings of size N, where the plaintexts have coefficients modulo t_p and ciphertexts modulo q_c. CRT encoded ciphertexts also allow for rotation of slots, which will be denoted as <<R or >>R in the figures, where R is a random integer that shows by how much the slots will be rotated, while the arrows show the direction of the rotation. For the pseudocodeswe will use the function Rotate([ctxt], R) which returns a rotation to the right (>>) for R slots of the ciphertext [ctxt], andIf R is a negative integer then the rotation is done for R slots to the left. If not stated otherwise, throughout the paper we assume that all of the encryptions are done by firstly encoding the plaintext messages to workin SIMD fashion and then encrypt them with a SWHE public scheme.

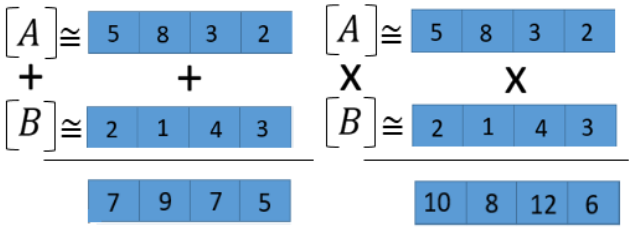

Figure 1. Illustration of SIMD regime of homomorphic operations.

System participants and their model

In our system we have 2 participant (entities): a **server** that has an already trained Naïve Bayes based model, and a client (user) that has an un-classified query (data) that he wishes to classify in privacy preserving fashion using the server's trained model (fig.1). This means that while being engaged in the privacy preserving protocol, the user hides both the un-classified query, the final prediction or any intermediate result from the server, while the server hides from the client all theparameters related to the trained model. We also assume that both the client and server are from the semi-honest (honest-but-curious) model, which means that they follow the protocol but on the background (while running the protocol) they try to infer some data which they are not

supposed to do. Motivations for parties to be from the semi-honest model and hence follow the protocol are given in [9-10].

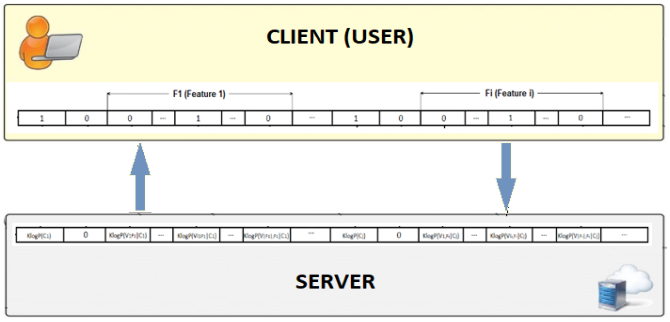

Figure 2. Participants (a server and a client) in the semi-honest model

BUILDING BLOCKS

In this section we will give some adoptable and flexible secure building blocks, which in turn can be used to build more complex algorithms in privacy preserving fashion

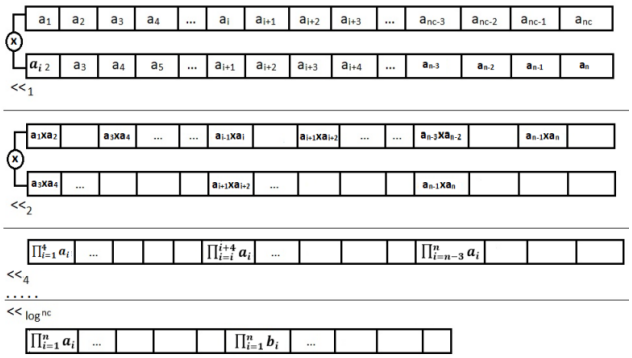

Figure 3. Illustration of the secProdOfBlocks algorithm.

Secure product of blocks of n slots

Allows to securely find the product of a block of n slots in a SIMD encoded ciphertext. Its illustration is given in fig.3, while its pseudocode is shown in algorithm 1.

ALGORITHM 1: secProductOfBlocks

INPUT: [inputCipher], n

n: size of the block of slots, starting from slot 0

OUTPUT: [result]

[result]: contains the product of each block of n slots at the beginning (first slot) of the corresponding block

1 [result] = [inputCipher]
2 for i = 1 to $\log_2(n)$
3 [result] = [result] x Rotate([result], -2^i) // fig.3
4 return [result]

Secure SIMD Comparison

It's a secure two party computation, where Party 1 has two encrypted ciphertexts [A] and [B] (not necessarily encrypted to enable SIMD), while Party 2 has the secret (decryption) key. In the end Party 2 learn which one is greater, but not by how much. Party 1 learns nothing. The SIMD version of the secure comparison algorithm is illustrated in fig.4., while the pseudocode is given in algorithm 2.

ALGORITHM 2: secComp

INPUT:

Party 1:[A], [B]– integers encrypted by Party 1's pub. key Pk

Party 2: secret key (sk) which can decrypt the ciphertexts

OUTPUT: result

result: If result ≥ 0 then A ≥ B, otherwise A< B

Party 1:
1 [C] = ([A] - [B]) x R // R > 0, fig.4
2 send [C] to Party 2
Party 2:
3result = Decrypt([C])

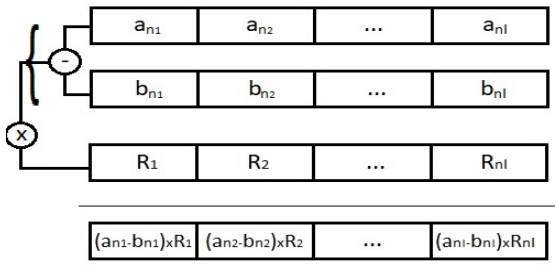

Fig.4. SIMD version secure comparison.

Secure argmax over encrypted data

In this scenario Party 1 has an array of c encrypted integers using the public key of Party 2. They are not necessarily encrypted to enable SIMD. Party 2 has the secret (decryption) key. In the end Party 2 learns only the index of the maximum integer of the array and nothing else (neither by how much the numbers in the array differ nor their relative order in term of which one is greater). Party 2 learns nothing. Algorithm 3 gives the pseudocode for the secure argmax protocol.

ALGORITHM 3: secArgmax

INPUT:

Party 1: [arr[]] - array of c integers, encrypted by P1's Pk

Party 2: secret key (sk) which can decrypt the ciphertexts

OUTPUT:maxIndex

maxIndex: the index of the maximum integer in array[]

Party 1:
1 permute the array using random permutation π(arr[])
2 compute all comparisons = $\{\pi(arr[x_i]) - \pi(arr[x_j])\}R_{ij}$

//for i = 1, …c and j=i+1, …n; $R_{ij}$is a random integer
3 send comparisons toParty 2
Party 2:
4receive the c(c-1)/2 comparisons combinations.
decrypt them.Find the maximum one and send {σ_1, … σ_c}
to Party2, where σ_i=[0] if π(array[xi]≠max, otherwise σ_i = [1]
5 send all σ_i in order (starting from σ_1 to σ_n) to Party 1
Party 1:
6compute $v = \sum_i \pi^{-1}(i) \cdot \sigma_i$ and send *v* to A
//π^{-1}(i) are the indices of the inverse permutation, not integers
7 send v to Party 2
Party 2:
8 maxIndex=Decrypt(v) //index of the max. integer of arr[]

Basically, in line 1 Party 1 permutes the array with a random permutation π, and in line 2 for each pair it does the secure comparison technique (as its described in algorithm 2) and sends all of the comparison to Party 2 (line 3). Party 2decrypt the results for all the pair comparisons (line 4), and having in mind the logic of algorithm 2 it finds the index of the maximum integer (the one for which all the comparisons yielded a result greater than zero) of the permuted array. Then it encrypts n integers such thatσ_i = [1] if i is the index of the maximum integer of the permuted array, otherwise σ_i = [0], for i=1, … c (line 4) and sends them to Party 1 again (line 5). Party A finds the encryption of the maxIndex of the original array arr[] by homomorphically executing $v = \sum_i \pi^{-1}(i) \cdot \sigma_i$, where π^{-1}(i) is the index value of the inverse permutation of π (line 6) and sends this v to Party 2 (line 7).Finally, Party 2 decrypts v to find the index of the integer with the biggest value in the original array arr[] (line 8).

SECURE NAÏVE BAYES CLASSIFICATION

Algorithm 4 gives the pseudocode for the privacy preserving Naïve Bayes classification. As an input the server has the trained model which includes all the class and the conditional value-class probabilities. The trained model consists of c plaintexts (one for each class) encoded to enable plain SIMD operations on them. Each of the classes' plaintext at the first slot has the probability of that class, followed by the class-value conditional probabilities of all values among all features in an ordered sequence (i.e. starting with the conditional probability of the first value of $F_1$with the class, then second value, and so on till the end). The same goes for other features. SotrainedModel = $\{P(C_k), P(V_{1,F1}|C_k), P(V_{2,F1}|C_j), \dots N(V_{|Ff|,Ff}|C_k)\}_{k=1}^{c}$.

ALGORITHM 4: PPNaiveBayesClassification

INPUT:
Server: trainedModel = {class_C_1, class_C_k} //fig.4
Client: secret key (sk), query feature vector

OUTPUT:finalClassification
finalClassific.: the predicted label of class C={C_1, C_2, … C_c}.

Client:
1 [queryFV] = Encrypt(query)
2 [mask] = Encrypt({1, 1, …, 1})
3 [inverseQFV] = [mask] - [queryFV]
4 send [queryFV] and [inverseQVF] to Server
Server:
5 for k = 1 to c
6 [res] = [queryFV] x class_C_k // fig.6
7 [res] = [res] + [inverseQVF] // fig.7
8 [Prob[k]] = secProductOfBlocks([res], $\sum_{i=1}^{f}|F_i| + 1$)
Server (as Party 1) and Client (as Party 2):
9 finalClassification= secArgmax (Prob[ ])

This is illustrated in fig.5. Since SWHE schemes don't work with real numbers, all of the probabilities are multiplied by a constant K and rounded to the closest integer number. For the Naïve Bayes case it has been shown that it is sufficient for this K to be between 8-10 bits so to have the same accuracy as the plain (non-privacy preserving) classifier [13].

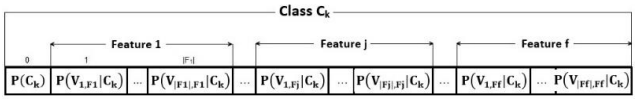

Figure 5. SIMD encoding of the probabilities related to class C_k.

On the other hand, besides the secret key, as an input the client also has the un-classified "query feature vector" that he wishes to classify. It is constructed as a single ciphertext that has one slots for each of the values of all of the features arranged in ordered sequence. If a certain value of a certain feature is part of the query we put 1 on the corresponding slot and all the other slots have value 0. The query feature vector is illustrated in fig.6 (the upper vector of fig.6).

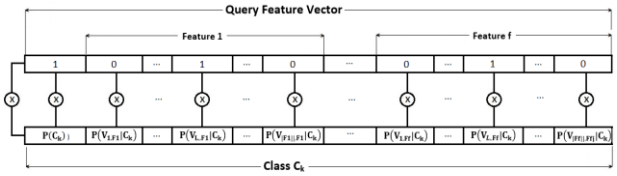

Figure 6. SIMD multiplication of the query feature vector with class C_k.

The client encrypts the query feature vector (line 1), and in order to enable the secProductOfBlocks algorithm (Algorithm 1), he also constructs the inverse query feature vector, which has the bits inverted (switched)

from the original query, i.e. ones become zero and zeros become ones. This is illustrated in fig.7 (lower vector of fig.7). This is done by first encrypting a vector of all ones (line 2), then subtracting it from the original vector (line 3). The client sends the both of them to the server (line 4).

Upon receiving both of the encrypted vectors, for each of the c classes the Server multiplies the received query with the class vector (line 6) as it is illustrated in fig.6, then adds the inverseQFV to the previous result (line 7), which is shown in fig.7. Afterwards, by calling Algorithm 1 it finds the likelihood of the query to belong to the corresponding class (line 8). We should note that the size of the blocks here is n = $\sum_{i=1}^{f}|F_i|+1)$ slots, i.e. one for all feature-values for all the features, and one for the class probability. At this point the server has found all of the c encrypted probabilitylikelihoods of the query to belong to acertainclass.Together with the client they engage in the argMax algorithm (Algorithm 3), at the end of which the client has the final Classification of his query (line 9). This was done by satisfying the privacy preserving classification goals set in section 1, for both the server and the client.

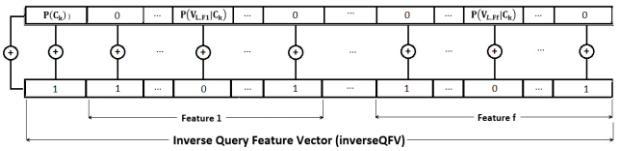

Figure 7. Adding the inverseQFV to the previous multiplication result.

We should also note that in order to increase the throughput and performances by a large amount of magnitude without any cost increase, we can pack several (say up to q) queries at the client. In turn this should be reciprocated by class packing at the server (each of c plaintexts of the trained model has the same class packed for q times) (fig.8). In this way Algorithm 4 is able to process q queries in one call.

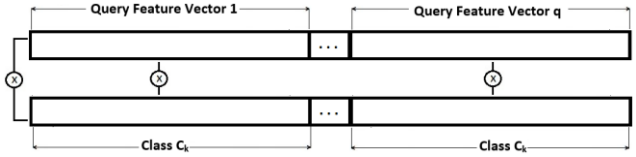

Figure 8. Query and class packing to increase throughput without extra cost.

EXPERIMENTAL RESULTS. COMPARISONS

In order to test the performance of our building blocks and scheme, we use the Wisconsin breast cancer dataset from the UCI repository [14]. It has 9 features (hence f=9), and two classes – cancer and non-cancer (thus c=2). All of the features can have values between 1 and 10 (hence |Fi|=10 for 1≤i ≤f). This means that for the block size of one class (fig. 5) we have $\sum_{i=1}^{f}|F_i|+1)$ =91 slots. For the purposes of Algorithm 1, we will round it up the closest power of two, which is n=128 = 2^7. This means that for each class we need to have 128 slots to encode it to work in SIMD fashion.

Table 1. Accuracy of Algorithm 4.

	Pred. non-cancer	**Pred. cancer**
True non-cancer	436	22
True cancer	6	235

For encryption purposes we use the Microsoft's SEAL library [15], which is based on the FV SWHE scheme [11]. Due to circuit depth (noise issues, see [11-15] for more details) for the polynomial size (modulus) we tried the values of N=4096, 8192 and 16384. It was shown that 25 bits were enough for the plain modulus t_p so not to have accuracy loss (table 1) due to the integerization process (multiplying by K). The coefficient modulus was chosen according to standardized parameters to adjust for a 128-bit security for all of the above polynomial sizes. We run the code on a Windows 10 platform with an Intel i5 processor of 2.4GHz and 4 GB of RAM. The accuracy was 96% (table 1) for both the privacy preserving and the plain classifier.

Fig.9 gives the timing results for both the client and the server. Y-axis represents the time in ms (milliseconds), while the X-axis is the average time per query throughput. Having in mind that a single class needs n=128 slots, then for N=4096, 8192 and 16384 we have a throughput of q=N/n = 32, 64 and 128 queries per ciphertext, correspondingly.

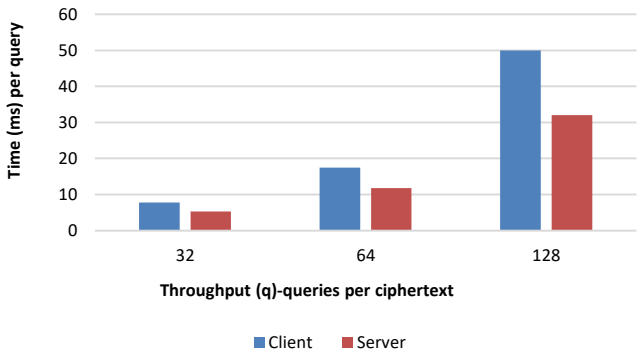

Figure 9. Average time in ms (y-axis) per query processing when the throughput is q = 32, 64, 128 queries per ciphertext (x-axis).

Table 2 provides some comparisons related to the performance and security characteristics for the building blocks of secure comparison and secure argmax among several privacy preserving schemes.

Table 2. Comparisons of secure comparison and argmax.

Scheme		Bost et. al. [6]	Sun et. al. [7]	Gao et. al. [8]	Liu et. al. [9]	Liu et. al. [10]	Our scheme
Sec. comp	**Nr. of rounds**	3 rounds	1 rounds	3 rounds	1 rounds	1 rounds	1 rounds
	Leakage	No Leakage	No Leakage	The difference between two number exposed if protocol run several times	No Leakage	No Leakage	No Leakage
argmax	**Nr. of rounds**	3 (c-1) rounds	c-1 rounds	Does not cover multiclass class.	c-1 rounds	Does not cover multiclass class.	2 rounds
	Leakage	No Leakage	No Leakage	Leaks the order (sequence) of the numbers, but not their values	No Leakage	No Leakage	No Leakage

Table 3. Results for cumulative classification per query.

PP Classification Scheme	Bost et. al. [6]	Sun et. al. [7]	Gao et. al. [8]	Liu et. al. [9]	Liu et. al. [10]	Our scheme
Computation	479 ms	48.79 ms	555 ms	348. 8 min.	196 sec.	14 ms
Communication	72.47 KB	Not reported	19.3 KB	1.244 MB	Not reported	109 KB

Table 3 gives the cumulative computation and communication cost for both the client and the server among several schemes. We should note that for experimental purposes our scheme uses the same dataset as [6], [9] and [10], while the other schemes use similar datasets in terms of number of features and number of values per feature set, which makes the schemes pretty much comparable to each other.

SECURITY ANALYSIS

We base our security on the hardness of the RLWE problem of our SWHE scheme, which also provides a semantic security [11]. This semantic security will also enable us to do the simulations for the real and simulated world when proving the security in the semi-honest model,for both our building blocksas well as our privacy preserving classification scheme, in a similar fashion as it is done in [6-10] and in almost all similar papers.

DISCUSSIONS AND CONCLUSION

In this paper we presented a few original secure building blocks which can be incorporated to build any privacy preserving algorithm. Utilizing them we constructed a privacy preserving Naïve Bayes classifier. Our experimental results, as well as theoretical analyses showed that our building block and scheme outperformed, many of the state-of-the-art ones in terms of computation and communication cost. Furthermore, our blocks and schemes showed to be more robust and have better properties and security characteristics than most of the others, but due to space concerns we omitted them here. We plan to address them in an eventual extension of this paper. For the upcoming research we plan to address the issue of privacy preserving classification for other algorithms such as deep learning, SVM, decision trees, random forests, etc.We also consider to extend this paper to address the issue of privacy preserving training as well.

REFERENCES

[1] Reinsel, David, John Gantz, and John Rydning. "Data age 2025: The evolution of data to life-critical." Don't Focus on Big Data (2017).

[2] Shokri, Reza, and VitalyShmatikov. "Privacy-preserving deep learning." Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. ACM, 2015

[3] Agrawal, Rakesh, and RamakrishnanSrikant. Privacy-preserving data mining. Vol. 29. No. 2. ACM, 2000.

[4] Lindell, Yehuda, and Benny Pinkas. "Privacy preserving data mining." Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 2000.

[5] Gilad-Bachrach, Ran, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy." In International Conference on Machine Learning, pp. 201-210. 2016

[6] Bost, Raphael, Raluca Ada Popa, Stephen Tu, and ShafiGoldwasser. "Machine learning classification over encrypted data." In NDSS. 2015

[7] Sun, Xiaoqiang, et al. "Private machine learning classification based on fully homomorphic encryption." IEEE Transactions on Emerging Topics in Computing (2018).

[8] Gao, Chong-zhi, et al. "Privacy-preserving Naive Bayes classifiers secure against the substitution-then-comparison attack." Information Sciences 444 (2018): 72-88.

[9] Liu, Ximeng, Robert Deng, Kim-Kwang Raymond Choo, and Yang Yang. "Privacy-Preserving Outsourced Clinical

Decision Support System in the Cloud." IEEE Transactions on Services Computing (2017).

[10] Liu, Ximeng, Rongxing Lu, Jianfeng Ma, Le Chen, and Baodong Qin. "Privacy-preserving patient-centric clinical decision support system on naive Bayesian classification." IEEE journal of biomedical and health informatics 20, no. 2 (2016): 655-668

[11] Fan, Junfeng, and Frederik Vercauteren. "Somewhat Practical Fully Homomorphic Encryption." IACR Cryptology ePrint Archive 2012 (2012): 144.

[12] Smart, Nigel P., and Frederik Vercauteren. "Fully homomorphic SIMD operations." Designs, codes and cryptography 71, no. 1 (2014): 57-81.

[13] Park, Heejin, Pyung Kim, Heeyoul Kim, Ki-Woong Park, and Younho Lee. "Efficient machine learning over encrypted data with non-interactive communication." Computer Standards & Interfaces 58 (2018): 87-108.

[14] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

[15] https://github.com/microsoft/SEAL

THIS PAGE INTENTIONALLY LEFT BLANK