A Privacy-Preserving Solution for the Bipartite Ranking Problem

Noushin Salek Faramarzi, Erman Ayday, H. Altay Güvenir
Computer Engineering Department, Bilkent University
Email: noushin.salek@bilkent.edu.tr, erman@cs.bilkent.edu.tr, guvenir@bilkent.edu.tr

***Abstract*—In this paper, we propose an efficient solution for the privacy-preserving of a bipartite ranking algorithm. The bipartite ranking problem can be considered as finding a function that ranks positive instances (in a dataset) higher than the negative ones. However, one common concern for all the existing schemes is the privacy of individuals in the dataset. That is, one (e.g., a researcher) needs to access the records of all individuals in the dataset in order to run the algorithm. This privacy concern puts limitations on the use of sensitive personal data for such analysis. The RIMARC (Ranking Instances by Maximizing Area under the ROC Curve) algorithm solves the bipartite ranking problem by learning a model to rank instances. As part of the model, it learns weights for each feature by analyzing the area under receiver operating characteristic (ROC) curve. RIMARC algorithm is shown to be more accurate and efficient than its counterparts. Thus, we use this algorithm as a building-block and provide a privacy-preserving version of the RIMARC algorithm using homomorphic encryption and secure multi-party computation. Our proposed algorithm lets a data owner outsource the storage and processing of its encrypted dataset to a semi-trusted cloud. Then, a researcher can get the results of his/her queries (to learn the ranking function) on the dataset by interacting with the cloud. During this process, neither the researcher nor the cloud learns any information about the raw dataset. We prove the security of the proposed algorithm and show its efficiency via experiments on real data.**

I. Introduction

The goal of privacy-preserving algorithms in data mining is to lower the risk of misuse of individuals' sensitive data and at the same time produce high quality results (i.e., similar to the ones that would be produced in the absence of privacy preserving techniques [1]. In this work, we focus on the bipartite ranking problem and we propose an algorithm that efficiently solves the problem in a privacy-preserving way.

Bipartite ranking problem is about learning a ranking function from a training set of positively and negatively labelled examples. Once the resulting ranking function is applied to a new (unlabelled) instance, the function is expected to establish a total order in which positive instances precede negative ones [2]. The most commonly used criterion for measuring the quality of the resulting bipartite ranking function is the area under the receiver operating characteristic (ROC) curve (referred as AUC).

One common drawback of the algorithms that solve the bipartite ranking problem is their applicability in real-life settings. This drawback arises due to privacy-sensitivity of personal data that is collected by the data owner. In most cases, the data owner (that collects and labels the data) does not have sufficient resources (i.e., storage and computation power) to answer the queries (to learn a ranking function on the dataset) of the researchers about the dataset. For instance, most researchers request sensitive medical information from hospitals to work on, but privacy concerns make the hospitals unwilling to provide such information.

On the one hand, getting the result of such queries (and learning the ranking function) is very valuable for the researchers in most cases. On the other hand, the data owner does not directly share its own dataset with the researcher due to the aforementioned privacy and legal concerns. Therefore, usually, the data owner has two options: (i) the data owner may anonymize the dataset before sharing it with the researchers, which reduces the utility (or accuracy) of the dataset, and hence the query result, or (ii) it can outsource the storage and processing of the dataset to a trusted party, however existence of such a trusted party is not practical in most real-life settings. In this work, we focus on the latter option, but rather than assuming the existence of a fully trusted party, we resort to using cryptographic techniques.

As discussed, there are many algorithms in the literature that solves the bipartite ranking problem [2], [3]. A recent algorithm, named RIMARC (Ranking Instances by Maximizing Area under the ROC Curve) achieves high AUC values compared to other works, while providing a low time complexity [4]. In this work, we use the RIMARC algorithm as a building-block and propose a privacy-preserving version of the RIMARC algorithm. To achieve our goal, we use homomorphic encryption and secure multi-party computation.

The proposed algorithm ensures that no party other than the data owner can access to the content of the database. Furthermore, the researchers can only obtain the results of their queries, and the cloud does not learn anything about the dataset. We prove the security of the proposed algorithm and show its efficiency via experiments on real data.

The rest of the paper is organized as follows. In the next section, we provide background information about the RIMARC algorithm and the cryptographic tools we use in our algorithm. In Section III, we describe our proposed privacy-preserving algorithm in detail and we also provide a brief security analysis. In Section IV, we provide the experimental results. Finally, in Section V, we conclude the paper and discuss potential future works.

978-1-5090-6167-9/16 $31.00 © 2016 IEEE
DOI 10.1109/ICMLA.2016.27

II. Background

Here, we provide brief backgrounds on the technical concepts and algorithms we use in this paper.

A. Receiver Operating Characteristic Curve

Receiver operating characteristic (ROC) curve is a well-known tool for evaluating the performance of binary classifiers. It is a powerful metric compared to traditional accuracy metrics. The ROC curve is generated by plotting the ratio of the *true positive rate* (TPR) to the *false positive rate* (FPR) at different threshold values. Each (FPR, TPR) pair corresponds to a specific point on the ROC curve. A perfect classifier yields a point on the upper left corner (or coordinate $(0, 1)$) of the ROC space. On the other hand, a completely random guess would give a point along a diagonal line from the left bottom to the top right corners towards $(0.5, 0.5)$.

B. Area Under the ROC Curve

The area under the ROC curve (AUC) is a measure of the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. The highest possible AUC value is 1.0 which represents a perfect classification, and a value of 0.5 corresponds to a random decision [5]. Therefore, the values below 0.5 can be easily neglected. A feature with a higher AUC value can determine the class label with a higher relevance. AUC is an indicator for quality of ranking and the higher AUC implies better ranking.

C. RIMARC Algorithm

Ranking Instances by Maximizing Area under the ROC Curve (RIMARC) algorithm learns a model that ranks instances based on how they are likely to have a positive label; an attempt for maximizing the AUC [4]. Each ranking function is learned for each feature in order to maximize the AUC. The AUC value, obtained for a single feature shows the effect of that feature in ranking. An important property of such a ranking function is that it is in a human readable form that can be easily assessed by domain experts.

In order to construct the ranking function for a given feature f_i, all continuous features are first discretized into categorical ones in a way that optimizes the AUC. This discretization can be done by a method called "MAD2C" [6]. The score value (S) for a given category j of a specific feature f_i, including discretized continuous features, can be computed as below:

$$S(c_i^j) = \frac{P(c_i^j)}{P(c_i^j) + N(c_i^j)} \tag{1}$$

Here, c_i^j represents the j^{th} category of feature f_i. Also, $P(c_i^j)$ and $N(c_i^j)$ represent the total number of positive and negative instances of c_i^j, respectively.

All the categories (of a given feature) are sorted according to their score values computed in the previous step. Since the ranking function used by RIMARC always results in a convex ROC curve, the AUC is always greater than or equal to 0.5. The ROC curve points, (FPR, TPR), corresponding to each score value is calculated at this step. Using these points, the AUC value is determined. The weight of a feature, f_i is computed as, $w_i = 2(AUC(i) - 0.5)$, where $AUC(i)$ is the AUC obtained for feature f_i.

Label	Color
N	R
P	G
N	B
N	W
P	Y
N	W
N	R
N	B
N	Y
P	G
N	W
P	G
P	G
N	R
N	Y

(a)

Category	Representation				
R	1	0	0	0	0
G	0	1	0	0	0
Y	0	0	1	0	0
B	0	0	0	1	0
W	0	0	0	0	1

(b)

Fig. 1. Toy example of the RIMARC algorithm. (a) training dataset including a single feature (color) with 5 categories. Labels "N" and "P" represent the negative and positive labels, respectively. (b) bit representations of the categories (this representation will be discussed in Section III).

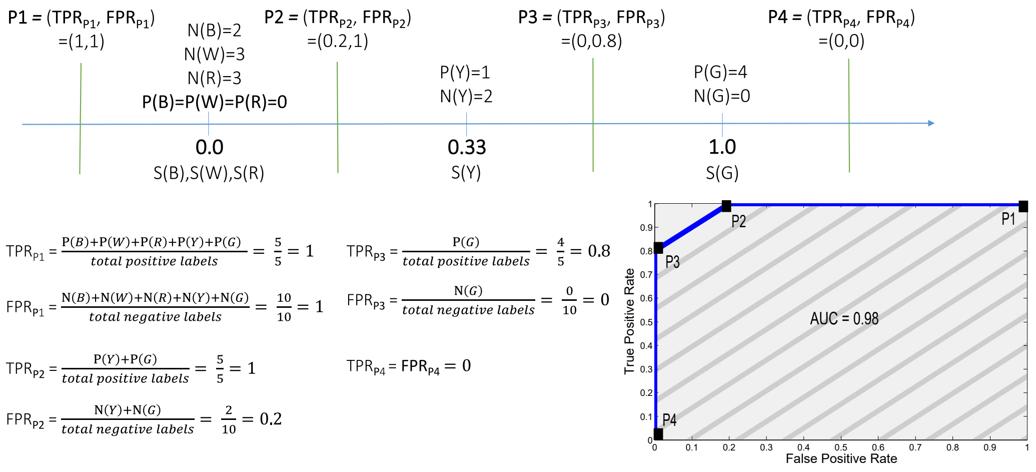
A toy example. To understand how RIMARC works, consider a toy training dataset with a single feature as shown in Fig. 1(a). The feature we consider in this example has a total of 5 categories (R, B, W, Y, G). The score values of these categories are obtained, using Eq. 1, as follows: $S(R) = S(B) = S(W) = 0$, $S(Y) = 0.33$, $S(G) = 1.0$. As shown in Fig. 2, the score values are sorted and mapped on an axis. Then, TPR and FPR values calculated for each score value. AUC value is determined using the area under the ROC curve.

D. Homomorphic Encryption

In this work, we use the Paillier cryptosystem [7] that provides additive homomorphism. Let p and q be two large prime numbers, $n = p \cdot q$ be the security parameter, g be a random integer (such that $g \in \mathbb{Z}_{n^2}^*$), and λ be the least common multiple of $(p-1)$ and $(q-1)$. Let also the modular multiplicative inverse $\mu = (\mathrm{L}(g^\lambda \mod n^2))^{-1}$, where $\mathrm{L}(u) = (u-1)/n$. Then, the public key pk is represented by the pair (n, g) and the private key sk is represented by the pair (λ, μ).

Encryption of a message m ($m \in \mathbb{Z}_n$) is done by selecting a random number r ($r \in \mathbb{Z}_n^*$) and computing the ciphertext $E(m) = g^m \cdot r^n \mod n^2$. *Decryption* of an encrypted message c ($c \in \mathbb{Z}_{n^2}^*$) is done by computing $D(c, sk) = \mathrm{L}(c^\lambda \mod n^2) \cdot \mu \mod n$.

The Paillier cryptosystem is an additively homomorphic cryptosystem and, as such, it supports some computations in the ciphertext domain. In particular, let m_1 and m_2 be two messages encrypted with the same public key pk. Then, the encryption of the sum of m_1 and m_2 can be computed as $E(m_1 + m_2) = E(m_1) \cdot E(m_2)$. Furthermore, any ciphertext $E(m)$ raised to a constant number c is equal to the encryption of the product of the corresponding plaintext and the constant as $E(m \cdot c) = E(m)^c$.


P1 = (TPR$_{P1}$, FPR$_{P1}$) =(1,1)
N(B)=2
N(W)=3
N(R)=3
P(B)=P(W)=P(R)=0
P2 = (TPR$_{P2}$, FPR$_{P2}$) =(0.2,1)
P(Y)=1
N(Y)=2
P3 = (TPR$_{P3}$, FPR$_{P3}$) =(0,0.8)
P(G)=4
N(G)=0
P4 = (TPR$_{P4}$, FPR$_{P4}$) =(0,0)
0.0
S(B),S(W),S(R)
0.33
S(Y)
1.0
S(G)
True Positive Rate
False Positive Rate
P1
P2
P3
P4
AUC = 0.98

$$TPR_{P1} = \frac{P(B)+P(W)+P(R)+P(Y)+P(G)}{total\ positive\ labels} = \frac{5}{5} = 1$$

$$FPR_{P1} = \frac{N(B)+N(W)+N(R)+N(Y)+N(G)}{total\ negative\ labels} = \frac{10}{10} = 1$$

$$TPR_{P2} = \frac{P(Y)+P(G)}{total\ positive\ labels} = \frac{5}{5} = 1$$

$$FPR_{P2} = \frac{N(Y)+N(G)}{total\ negative\ labels} = \frac{2}{10} = 0.2$$

$$TPR_{P3} = \frac{P(G)}{total\ positive\ labels} = \frac{4}{5} = 0.8$$

$$FPR_{P3} = \frac{N(G)}{total\ negative\ labels} = \frac{0}{10} = 0$$

$$TPR_{P4} = FPR_{P4} = 0$$

Fig. 2. Calculation of the TPR and FPR values for the toy example (in Fig. 1(a)). Score values of the categories are determined using Eq. 1. Thus, $S(R) = S(B) = S(W) = 0$, $S(Y) = 0.33$, and $S(G) = 1.0$. TPR and FPR values are computed for each score value as shown in the figure. Then, the AUC value is computed in the graph based on the computed (FPR,TPR) points as (P1,P2,P3,P4).

III. Proposed Solution

In this section, our solution for privacy preserving RIMARC algorithm is explained in detail.

A. System and Threat Models

We have three main parties in the system: (i) data owner, which collects and provides the dataset that consists of sensitive data, (ii) cloud, which is responsible for the storage and processing of the dataset, and (iii) researcher, which is interested in analyzing the dataset and obtaining ranking functions out of it.

Dataset is stored at the cloud in encrypted form (details are provided in the next subsections). Our goal is to make sure that no party other than the data owner can access the plaintext (non-encrypted) format of the whole dataset. We also want to make sure that (i) the cloud does not learn any information about the dataset (including the result that is provided to the researcher), and (ii) the researcher only learns the results of his (authorized) query, and nothing else. In order to achieve these goals, we propose a privacy-preserving algorithm between the cloud and the researcher in order to compute that AUC values. Our proposed protocol involves cryptographic primitives such as homomorphic encryption and secure two-party computation.

We assume the data owner to have public/private key pair (pk_o, sk_o) for the Paillier cryptosystem (as discussed in Section II-D). Public key of the data owner is known by all parties in the system and the secret key is only shared with the researcher (so that the researcher can decrypt and obtain the results of his queries). Note that, rather than providing the private key of of the data owner to the researcher, it is also possible to use a threshold cryptography scheme, in which the private key is divided into two parts and these parts are distributed to the researcher and the cloud [8]. This threshold cryptography feature can be easily integrated into the proposed algorithm depending on the use case scenario. Our proposed system model is also illustrated in Fig. 3.

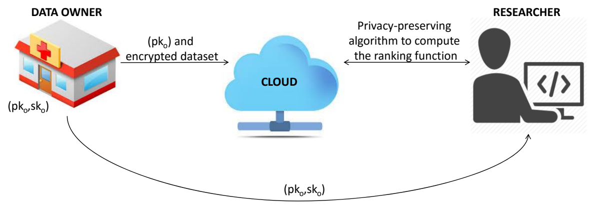

DATA OWNER
(pk$_o$,sk$_o$)
(pk$_o$) and encrypted dataset
CLOUD
Privacy-preserving algorithm to compute the ranking function
RESEARCHER
(pk$_o$,sk$_o$)

Fig. 3. Proposed system model.

We assume all parties in the system to be honest-but-curious. That means both the cloud and the researcher can try to learn the sensitive data of the data owner (i.e., the content of the database), but they honestly follow the protocol steps. We also assume that the researcher and the cloud does not collude during the protocol. Of course, not every researcher can send queries to the dataset; the researcher should be authorized in order to do so. This can be managed via an access control mechanism, which is out of the scope of this work. Finally, we assume all communications between the parties to be secured via end-to-end encryption.

B. Dataset Format and Encryption

Initially, the data owner collects data from record owners and constructs the dataset. For instance, this can be considered as a hospital collecting data from patients. We assume that the dataset consists of the records of N individuals (e.g., N patients)[1]. Set of features in the dataset is illustrated as $\mathbb{F} = \{f_1, f_2, \ldots, f_n\}$ (we assume there are n features). Also, set of categories for a given feature f_i is represented as $\mathbb{C}_i =$

[1]We also assume that continuous features are first discretized into categorical ones.

$\{c_i^1, c_i^2, \ldots, c_i^k\}$. Here, we assume there are k categories per feature for simplicity, but each feature may also have different number of categories as well. Each category also has a label ℓ_i^j, where $\ell_i^j \in \{0,1\}$[2], $i \in \mathbb{F}$, and $j \in \mathbb{C}_i$.

We represent a category c_i^j as below:

$$c_i^j = b_i^{j,1}||b_i^{j,2}||\ldots||b_i^{j,j}||\ldots||b_i^{j,k},$$

where $b_i^{j,m} = 0$ when $m \neq j$ and $b_i^{j,m} = 1$ when $m = j$. For example, the categories of color feature in our toy example (discussed in Section II-C) and their corresponding bit representations are illustrated in Fig. 1(b). In this example, we have 5 categories (i.e., colors) for the color feature, and hence we use 5 bits in the representation, each assigned for a particular color. We use Paillier cryptosystem to encrypt the whole dataset. In a nutshell, the data owner encrypts all categories and corresponding labels using its public key (pk_o). To encrypt a category c_i^j, the owner encrypts all its bits individually. Thus, we represent the encryption of c_i^j as $[b_i^{j,1}]||[b_i^{j,2}]||\ldots||[b_i^{j,k}]$.[3] The owner also encrypts all labels ℓ_i^j to obtain the corresponding $[\ell_i^j]$ values. After encryption, the data owner sends the encrypted data to the cloud for storage.

C. Privacy-Preserving RIMARC Algorithm

In the following, for the simplicity of the presentation, we describe the proposed algorithm for a single feature f_i with k categories. Note that the algorithm can be easily generalized to handle multiple features (as we also show in our evaluations in Section IV). The main steps of the proposed solution are also illustrated in Fig. 4.

Initially, the cloud counts the number of instances of each category in the dataset. To compute the sum of instances for the category c_i^j, the cloud computes $[T(c_i^j)] = \sum_{m=1}^{N}[b_i^{m,j}]$. This summation can be easily carried out by using the homomorphic properties of the Paillier cryptosystem (as discussed in Section II-D).

Then, the cloud counts the number of positive labelled instances for each category. To do so, for each category c_i^j, the cloud needs to compute $[P(c_i^j)] = \sum_{m=1}^{N}[b_i^{m,j}][\ell_i^m]$. This computation, however cannot be carried out by using the homomorphic properties of the Paillier cryptosystem (as the homomorphic properties does not support multiplication of two encrypted messages). Therefore, we propose using the "secure multiplication algorithm" (in Algorithm 1) between the cloud and the researcher in order to handle this multiplication in a privacy-preserving way.

We note that step 7 of Algorithm 1 is to remove the noise from the product and it can be easily done at the cloud using the homomorphic properties of the Paillier cryptosystem. We discuss the security of the secure multiplication algorithm in Section III-D. Similarly, the cloud also computes the counts for number of negative labelled instances for each category.

[2]$\ell_i^j = 0$ represents a negative instance and $\ell_i^j = 1$ represents a positive instance.

[3]In the rest of the paper, we use angled brackets to represent the encryption of a message via Paillier encryption under the public key of the data owner.

Algorithm 1 Secure Multiplication Algorithm

```
Input: @Cloud: encrypted messages $[a]$ and $[b]$. @Researcher: $(pk_o, sk_o)$.
Output: @Cloud: $[a \times b]$. @Reseacher: $\perp$.
1: The cloud generates two random numbers $r_1$ and $r_2$ from the set {1,...,K}.
2: The cloud masks $[a]$ and $[b]$:
       $[\hat{a}] \leftarrow [a] \times [-r_1] = [a - r_1]$,
       $[\hat{b}] \leftarrow [b] \times [-r_2] = [b - r_2]$.
3: The cloud sends $[\hat{a}]$ and $[\hat{b}]$ to the researcher.
4: The researcher decrypts $[\hat{a}]$ and $[\hat{b}]$ with $sk_o$:
       $\hat{a} \leftarrow D([\hat{a}], sk_o)$,
       $\hat{b} \leftarrow D([\hat{b}], sk_o)$.
5: The researcher computes $\hat{a} \times \hat{b}$
6: The researcher encrypts the results with $pk_o$ to get $[\hat{a} \times \hat{b}]$ and sends it to the cloud.
7: The cloud computes $[a \times b] \leftarrow [\hat{a} \times \hat{b}] + [a]^{r_2} + [b]^{r_1} + [-r_1 \times r_2]$.
```

To do so, for each category c_i^j, the cloud needs to compute $[N(c_i^j)] = [T(c_i^j)] - [P(c_i^j)]$. Note that, this computation can be easily handled at the cloud using the homomorphic properties of the Paillier cryptosystem.

Next, the cloud computes the score value for each category. As discussed, the encrypted score value for a category c_i^j is computed as $[S(c_i^j)] = [P(c_i^j)]/[T(c_i^j)]$. Since Paillier cryptosystem does not support division of two encrypted numbers, we propose normalizing the score value of each category. For this normalization, we compute the following normalization constant for each category c_i^j:

$$[Z_i^j] = \prod_{\substack{m=1 \\ m \neq j}}^{k} [T(c_i^m)]. \tag{2}$$

Since this computation requires multiplication of encrypted messages, we use the secure multiplication protocol in Algorithm 1 (i.e., we iteratively multiply pairwise values). Then, the cloud computes the normalized score value for each category c_i^j as $[\hat{S}(c_i^j)] = [P(c_i^j)] \times [Z_i^j]$. We emphasize that we only need the order relationship between the score values of categories to compute the AUC values. Therefore, this normalization does not decrease the accuracy of the algorithm.

The computed normalized score values ($[\hat{S}(c_i^j)]$) need to be sorted in order to compute the AUC values. This sorting operation can be securely carried on between the cloud and the researcher by using a "pairwise secure comparison algorithm" [9]. This interactive algorithm compares two encrypted numbers and returns an encrypted value (to the cloud) indicating which value is greater. However, in our scenario, a feature may include tens of categories. Thus, such a pairwise comparison algorithm requires hundreds of pairwise comparisons, significantly reducing the efficiency of the algorithm. Therefore, for this step, we propose a more lightweight algorithm which only leaks the order of the normalized score values to the cloud. We describe this lightweight sorting algorithm in the following.

The cloud initially selects a random number r from the set $\{1, \ldots, K\}$ and encrypts it via pk_o to get $[r]$. Then, for

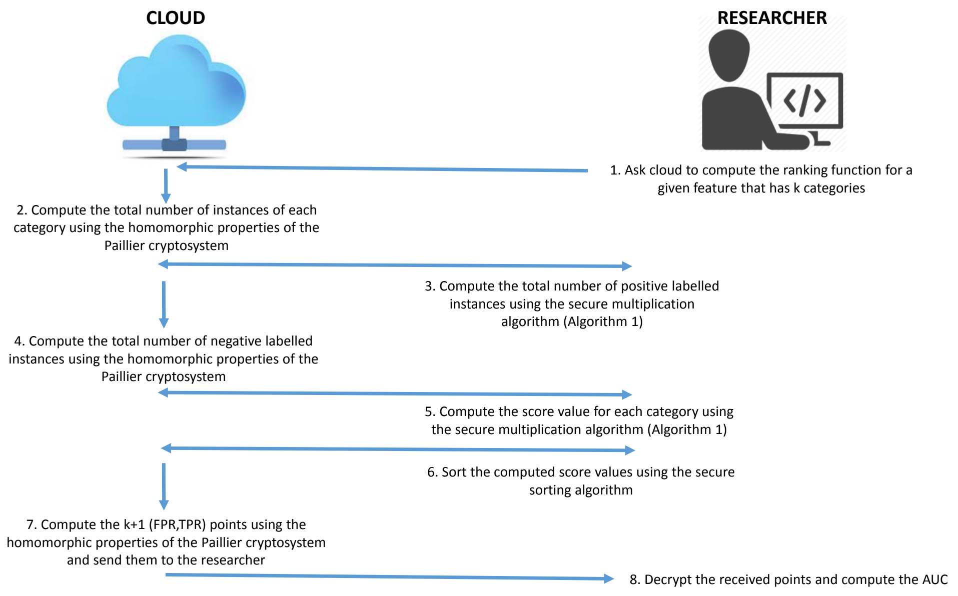

CLOUD
RESEARCHER
1. Ask cloud to compute the ranking function for a given feature that has k categories
2. Compute the total number of instances of each category using the homomorphic properties of the Paillier cryptosystem
3. Compute the total number of positive labelled instances using the secure multiplication algorithm (Algorithm 1)
4. Compute the total number of negative labelled instances using the homomorphic properties of the Paillier cryptosystem
5. Compute the score value for each category using the secure multiplication algorithm (Algorithm 1)
6. Sort the computed score values using the secure sorting algorithm
7. Compute the k+1 (FPR,TPR) points using the homomorphic properties of the Paillier cryptosystem and send them to the researcher
8. Decrypt the received points and compute the AUC

Fig. 4. Overview of the proposed solution. Steps 3, 5, and 6 are interactive steps between the cloud and the researcher.

each normalized score value $[\hat{S}(c_i^j)]$, the cloud computes the masked score value $[\tilde{S}(c_i^j)] = [\hat{S}(c_i^j)] + [r]$. Let $g(.)$ be a random shuffling function. The cloud shuffles the order of the masked score values as $g([\tilde{S}(c_i^1)], [\tilde{S}(c_i^2)], \ldots, [\tilde{S}(c_i^k)])$ and sends the masked (and encrypted) score values to the researcher in the shuffled order. The researcher decrypts the received score values, decrypts them using sk_o, and sorts them.

Since all normalized score values are masked with the same random number, their masked order is the same as the unmasked one. Therefore, the researcher can compute the correct sorting of these values. After the values are sorted at the researcher, he/she only sends the sorted order of the masked score values back to the cloud. Using this information and the shuffling order of the score values (which is known by the cloud), the cloud obtains the sorted order of the score values of all categories. We briefly discuss the security of this sorting algorithm in Section III-D.

The researcher only needs the $k+1$ (FPR, TPR) points to generate the AUC curve[4]. Knowing the order of $[\hat{S}(c_i^j)]$ values, the cloud can easily compute the $k+1$ encrypted ([FPR],[TPR]) points by using the number of positive and negative labelled instances for each category (i.e., $[P(c_i^j)]$ and $[N(c_i^j)]$ values) and by using the homomorphic properties of the Paillier cryptosystem (computation of the (FPR, TPR) points are discussed in Section II-C). Finally, the cloud sends the $k+1$ encrypted points to the researcher, the researcher decrypts the received values by using sk_o, and constructs the AUC curve. Note that TPR is the fraction of the true positives to the total positive labelled instances, and FPR is the fraction of the false positives to the total negative labelled instances. Since, this division cannot be carried out at the cloud, the cloud sends the numerator and denominator of each point to the researcher and the division is done at the researcher after the decryption.

D. Security Evaluation

Throughout the proposed algorithm, the cloud does not learn anything about the dataset of the data owner, and the researcher only learns the (FPR, TPR) points to generate the AUC curve.

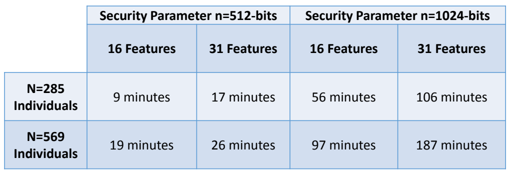
The proposed algorithm preserves the privacy of data owner's data relying on the security strength of the Paillier cryptosystem. The extensive security evaluation of the Paillier cryptosystem can be found in [7]. In particular, it is proved that the cryptosystem provides one-wayness (based on the composite residuosity class problem) and semantic security[5] (based on the decisional composite residuosity assumption).

We also use two interactive algorithms between the cloud and the researcher: (i) secure multiplication, and (ii) sorting. We briefly comment on their security in the following. During the secure multiplication algorithm, the cloud only gets the encrypted multiplication value, and hence computes $[P(c_i^j)]$. On the other hand, the researcher (due to the masking with random numbers r_1 and r_2) cannot observe any information about the labels or the categories.

During the sorting algorithm, since the values are masked and the researcher does not know the random number selected by the cloud, the researcher cannot learn the actual normalized score values. Furthermore, since the score values are shuffled

[4]Since we assume there are k categories, the AUC curve has totally $k+1$ points.

[5]The adversary cannot distinguish between two different encryptions of the same message.



	Security Parameter n=512-bits		Security Parameter n=1024-bits	
	16 Features	31 Features	16 Features	31 Features
N=285 Individuals	9 minutes	17 minutes	56 minutes	106 minutes
N=569 Individuals	19 minutes	26 minutes	97 minutes	187 minutes

TABLE I
TIME COMPLEXITY OF THE PROPOSED ALGORITHM FOR DIFFERENT SIZES OF THE SECURITY PARAMETER (n), DIFFERENT NUMBER OF FEATURES IN THE DATASET, AND DIFFERENT DATABASE SIZES.

at the cloud, the researcher cannot also learn which category has the highest score value. The cloud only learns the order of the normalized score values of the categories. Note that the data owner may prefer to keep the link between the bit encoding of a category and the name of the category hidden from the cloud (as the cloud does not need this information to do its operations). In such a scenario, learning the order relationship between the normalized score values does not mean anything to the cloud. Also, the selected random number r (from the set $\{1, \dots, K\}$) may be 0 with some probability. In this case, the researcher may learn the actual values of the normalized score values. However, this probability is negligibly small, especially for larger values of K.

IV. EVALUATION

In this section, we implement and evaluate the proposed algorithm on a real dataset. We used Wisconsin Diagnostic Breast Cancer (WDBC) dataset from UCI repository [10]. Dataset contains 569 individuals, each consisting of 31 features and labeled as either"M" for malignant (i.e., positive) and "B" for benign (i.e., negative). Also, each feature has different number of categories ranging from 8 to 19. As discussed in section II-C, we used MAD2C algorithm for this step.

To evaluate the practicality of the proposed algorithm, we implemented it and assessed its storage requirement and computational complexity on Intel Core i5-2430M CPU with 2.40 GHz processor under Windows 7. Our implementation is in Java and it relies on the MySQL 5.5 database. We set the size of the security parameter n for the Paillier cryptosystem to 512 and 1024 bits in different experiments. In Table I, we summarize the time complexity of the proposed algorithm.

Based on the results, we can say that the time complexity of the algorithm increases linearly with the number of individuals and features. Total time to run the privacy-preserving algorithm is on the order of minutes. Furthermore, the proposed algorithm is highly parallelizable as computations on each feature can be carried out independently. We will work on the parallelization of the algorithm in future work and we expect an improvement of at least one order of magnitude in this way. In terms of storage, the original (non-encrypted) dataset requires 120KB of storage space, while the encrypted version (thought the proposed algorithm) requires a disk space of 30MB when $n = 512$-bits and 60MB when $n = 1024$-bits. This increase in the storage requirement is due to the ciphertext expansion of the Paillier encryption. There exists techniques to reduce this ciphertext expansion (such as packing [11]), and we will utilize these techniques to optimize the storage requirement in future work.

Note that non-private version of the algorithm runs on the order of seconds and requires less storage space. However, we believe that this increase in time and storage complexity is a reasonable compromise in order to obtain a privacy-preserving algorithm, which is a crucial requirement to process personal data without privacy and legal concerns. We emphasize that the accuracy of the results (i.e., obtained ranking function) for the private algorithm is exactly the same as the non-private version. To summarize, the performance numbers we obtained show the practicality of our privacy-preserving algorithm.

V. CONCLUSION AND FUTURE WORK

In this work, we have developed a privacy-preserving solution for the bipartite ranking problem. We have developed an efficient and privacy-preserving version of the RIMARC algorithm. We have used cryptographic tools such as homomorphic encryption and secure multi-party computation to achieve our goals. Then, any researcher can mine the sensitive data by interacting with the cloud. The proposed algorithm guarantees that the cloud does not learn any information about the sensitive data, while the researcher only learns the result of his query. We have also showed the efficiency of the proposed algorithm via implementation using a real-life dataset. For future work, we will optimize the storage requirements of the proposed algorithm by using the packing technique. We will also further reduce the time complexity of the proposed algorithm by parallelizing our implementation for each feature.

REFERENCES

[1] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in *Annual International Cryptology Conference*. Springer, 2000, pp. 36–54.

[2] W. Kotlowski, K. J. Dembczynski, and E. Huellermeier, "Bipartite ranking through minimization of univariate loss," in *Proceedings of ICML*, 2011, pp. 1113–1120.

[3] M. R. Amini, T. V. Truong, and C. Goutte, "A boosting algorithm for learning bipartite ranking functions with partially labeled data," in *Proceedings of ACM SIGIR*, 2008, pp. 99–106.

[4] H. A. Güvenir and M. Kurtcephe, "Ranking instances by maximizing the area under roc curve," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 10, pp. 2356–2366, 2013.

[5] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[6] M. Kurtcephe and H. A. Güvenir, "A discretization method based on maximizing the area under receiver operating characteristic curve," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 27, no. 01, p. 1350002, 2013.

[7] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1999, pp. 223–238.

[8] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1–30, 2006.

[9] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2009, pp. 235–253.

[10] M. Lichman, "UCI machine learning repository," 2013.

[11] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Generating private recommendations efficiently using homomorphic encryption and data packing," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1053–1066, 2012.