ppAURORA: Privacy Preserving Area Under Receiver Operating Characteristic and Precision-Recall Curves

Ali Burak Ünal[1,3(✉)], Nico Pfeifer[2,3], and Mete Akgün[1,3]

[1] Medical Data Privacy Preserving Machine Learning (MDPPML), University of Tübingen, Tübingen, Germany
[2] Methods in Medical Informatics, University of Tübingen, Tübingen, Germany
[3] Institute for Bioinformatics and Medical Informatics (IBMI), University of Tübingen, Tübingen, Germany
{ali-burak.uenal,nico.pfeifer,mete.akguen}@uni-tuebingen.de

Abstract. Computing an area under the curve (AUC) as a performance measure to compare the quality of different machine learning models is one of the final steps of many research projects. Many of these methods are trained on privacy-sensitive data and there are several different approaches like ϵ-differential privacy, federated learning and cryptography if the datasets cannot be shared or used jointly at one place for training and/or testing. In this setting, it can also be a problem to compute the global AUC, since the labels might also contain privacy-sensitive information. There have been approaches based on ϵ-differential privacy to address this problem, but to the best of our knowledge, no exact privacy preserving solution has been introduced. In this paper, we propose an MPC-based solution, called ppAURORA, with private merging of individually sorted lists from multiple sources to compute the exact AUC as one could obtain on the pooled original test samples. With ppAURORA, the computation of the exact area under precision-recall and receiver operating characteristic curves is possible even when ties between prediction confidence values exist. We use ppAURORA to evaluate two different models predicting acute myeloid leukemia therapy response and heart disease, respectively. We also assess its scalability via synthetic data experiments. All these experiments show that we efficiently and privately compute the exact same AUC with both evaluation metrics as one can obtain on the pooled test samples in plaintext according to the semi-honest adversary setting.

Keywords: Privacy preserving AUC · ROC curve · PR curve · MPC

1 Introduction

Recently, privacy preserving machine learning studies aimed at protecting sensitive information during training and/or testing of a model in scenarios where data is distributed between different sources and cannot be shared in plaintext

S. Li et al. (Eds.): NSS 2023, LNCS 13983, pp. 265–280, 2023.
https://doi.org/10.1007/978-3-031-39828-5_15

[3,7,8,12,13,15,17,18]. However, privacy protection in the computation of the area under curve (AUC), which is one of the most preferred methods to compare different machine learning models with binary outcome, has not been addressed sufficiently. There are several differential privacy based approaches in the literature for computing the receiver operating characteristic (ROC) curve [2,5,6]. Briefly, they aim to protect the privacy of the data by introducing noise into the computation so that one cannot obtain the original data used in the computation. However, due to the nature of differential privacy, the resulting AUC is different from the one which could be obtained using non-perturbed prediction confidence values (PCVs) when noise is added to the PCVs [16]. For the precision-recall (PR) curve, there even exists no such studies in the literature. As a general statement, private computation of the exact AUC has never been addressed before to the best of our knowledge.

In this paper, we propose a 3-party computation based **p**rivacy **p**reserving **a**rea **u**nder **r**eceiver **o**perating characteristic and precision-**rec**a**l**l curves (ppAURORA). For this purpose, we use CECILIA [17] offering several efficient privacy preserving operations. The most important missing operation of it is division. To address the necessity of an efficient, private and secure computation of the exact AUC, we adapt the division operation of SecureNN [18]. Since the building blocks of CECILIA require less communication rounds than SecureNN, we implemented the division operation of SecureNN using the building blocks of CECILIA. Using ppAURORA, we compute the area under the PR curve (AUPR) and ROC curve (AUROC). We address two different cases of ROC curve in ppAURORA by two different versions of AUROC computation. The first one is designed for the computation of the exact AUC using PCVs with no tie. In case of a tie of PCVs of samples from different classes, this version just approximates the metric based on the order of the samples, having a problem when values of both axes of ROC curve plot change at the same time. To compute the exact AUC even in case of a tie, we introduce the second version of AUROC with a slightly higher communication cost than the first approach. Along with the privacy of the resulting AUC, since the labels are also kept secret during the whole computation, both versions are capable of protecting the information of the number of samples belonging to the classes from all participants of the computation. Otherwise, such information could have been used to obtain the order of the labels of the PCVs [19]. Furthermore, since we do not provide the data sources with the ROC curve, they cannot regenerate the underlying true data. Therefore, both versions are secure against such attacks [11]. We used the with-tie version of AUROC computation to compute the AUPR since the values of both axes can change at the same time even if there is no tie. To the best of our knowledge, ppAURORA is the first study for the privacy preserving AUPR computation.

2 Motivation

ppAURORA can enable collaborative privacy preserving evaluation of a binary model. Especially when there are parties with insufficient test samples, even

if these parties obtain the collaboratively trained model, they cannot reliably evaluate the predictions of this model on their test samples. The result of AUC on such a small set of test samples could vary significantly as shown in Fig. 1, making the reliability of the model evaluation questionable.

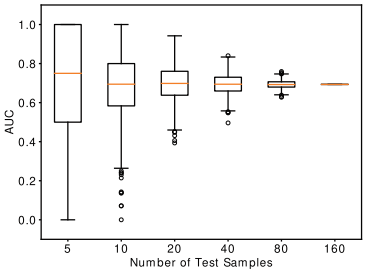

Fig. 1. AUROC for varying number of test samples from the all dataset

To demonstrate ppAURORA's contribute to the community more, let us imagine a scenario where a model is trained collaboratively using MPC framework [7,13,18], federated learning framework [9] or any other privacy preserving training method. Once the model is obtained, the participating parties can perform predictions on the model using their test samples to evaluate it. However, the parties with fewer data cannot reliably determine the performance of the model. Instead of individual evaluation of the model that could lead to incorrect assessment of the model's performance, they can use ppAURORA to evaluate it collaboratively and obtain the result of this evaluation as if it was performed on the pooled test samples of the parties without sacrificing the privacy of neither the labels nor the predictions of the samples.

3 Preliminaries

Security Model: In this study, we aim to protect the privacy of the PCVs and the labels of the samples from parties, the ranking of these samples in the globally sorted list and the resulting AUC. We prove the full security of our solution (i.e., privacy and correctness) in the presence of semi-honest adversaries that follow the protocol specification, but try to learn information from the execution of the protocol. We consider a scenario where a semi-honest adversary corrupts a single server and an arbitrary number of data owners in the simulation paradigm [4,10] where two worlds are defined: the real world where parties run the protocol without any trusted party, and the ideal world where parties make the computation through a trusted party. Security is modeled as the view of an adversary called a simulator $\mathcal{S}$ in the ideal world, who cannot be distinguished from the view of an adversary $\mathcal{A}$ in the real world. The universal composability framework [4] introduces an adversarial entity called environment $\mathcal{Z}$, which gives inputs to

all parties and reads outputs from them. The environment is used in modeling the security of end-to-end protocols where several secure protocols are used arbitrarily. Security here is modeled as *no environment can distinguish if it interacts with the real world and the adversary $\mathcal{A}$ or the ideal world and the simulator $\mathcal{S}$*. We also provide privacy in the presence of a malicious adversary corrupting any single server, which is formalized in [1]. The privacy is formalized by saying that a malicious party arbitrarily deviating from the protocol description, cannot learn anything about the inputs and outputs of the honest parties.

Notations: In our secure protocols, we use additive secret sharing over the ring $\mathbb{Z}_K$ where $K = 2^{64}$ to benefit from the natural modulo of CPUs of most modern computers. We denote two shares of x over $\mathbb{Z}_K$ with $(\langle x \rangle_0, \langle x \rangle_1)$.

CECILIA: In ppAURORA, we use secure multi-party computation framework CECILIA, which has three computing parties, P_0, P_1 and P_2, and uses 2-out-of-2 additive secret sharing where an ℓ-bit value x is shared additively in a ring among P_0 and P_1 as the sum of two values. For ℓ-bit secret sharing of x, we have $\langle x \rangle_0 + \langle x \rangle_1 \equiv x \mod L$ where P_i knows only $\langle x \rangle_i$ and $i \in \{0, 1\}$. All arithmetic operations are performed in the ring $\mathbb{Z}_L$.

3.1 Area Under Curve

One of the most common ways summarizing the plot-based model evaluation metrics is area under curve (AUC). It calculates the area under the curve of a plot-based model such as ROC curve and the PR curve.

Area Under ROC Curve (AUROC): The ROC curve takes the sensitivity and the specificity of a binary classifier into account by plotting the false positive rate (FPR) on the x-axis and the true positive rate (TPR) on the y-axis. AUC summarizes this plot by measuring the area between the line and the x-axis, which is the area under the ROC curve (AUROC). Let M be the number of test samples, $V \in [0, 1]^M$ contain the sorted PCVs of test samples in descending order, $T \in [0, 1]^M$ and $F \in [0, 1]^M$ contain the corresponding TPR and FPR values, respectively, where the threshold for entry i is set to $V[i]$, and $T[0] = F[0] = 0$. In case there is no tie in V, the privacy-friendly AUROC computation is as follows:

$$AUROC = \sum_{i=1}^{M} \Big(T[i] \cdot (F[i] - F[i-1]) \Big) \tag{1}$$

This formula just approximates the exact AUROC in case of a tie in V depending on samples' order. As an extreme example, let V have 10 samples with the same PCV. Let the first 5 samples have label 1 and the last 5 samples have label 0. Such a setting outputs $AUROC = 1$ via Eq. 1. In the reverse order, however, it gives $AUROC = 0$. To define an accurate formula for the AUROC in case of such a tie condition, let ξ be the vector of indices in ascending order where the PCV of the sample at that index and the preceding one are different

for $0 < |\xi| \leq M$ where $|\xi|$ denotes the size of the vector. Assuming that $\xi[0] = 0$, the computation of AUROC in case of a tie can be done as follows:

$$AUROC = \sum_{i=1}^{|\xi|} \Big(T[\xi[i-1]] \cdot (F[\xi[i]] - F[\xi[i-1]]) + \frac{(T[\xi[i]] - T[\xi[i-1]]) \cdot (F[\xi[i]] - F[\xi[i-1]])}{2} \Big) \tag{2}$$

As Eq. 2 indicates, one only needs TPR and FPR values on the points where the PCV changes to obtain the exact AUROC. We will benefit from this observation in the privacy preserving AUROC computation when there is a tie condition in the PCVs.

Area Under PR Curve (AUPR): The PR curve evaluates binary models by plotting recall on the x-axis and precision on the y-axis and summarizes it by measuring the area under the PR curve (AUPR). It is generally preferred over AUROC for problems with class imbalances. Since both precision and recall can change at the same time even without a tie, we measure the area by using the Eq. 2 where T becomes the precision and F becomes the recall.

4 ppAURORA

In this section, we give the description of our protocol for ppAURORA where we have data owners that outsource their PCVs and the ground truth labels in secret shared form and three non-colluding servers that perform 3-party computation on secret shared PCVs to compute the AUC. The data sources start the protocol by outsourcing the labels and the predictions of their test samples to the servers. Afterward, the servers perform the desired calculation privately. Finally, they send the shares of the result back to the data sources. The communication between all parties is performed over a secure channel (e.g., TLS).

Outsourcing: At the start of ppAURORA, each data owner H_i has a list of PCVs and corresponding ground truth labels for $i \in \{1, \ldots, n\}$. Then, each data owner H_i sorts its whole list L_i according to PCVs in descending order, divides it into two additive shares L_{i_0} and L_{i_1}, and sends them to P_0 and P_1, respectively. We refer to P_0 and P_1 as *proxies*.

Sorting: After the outsourcing phase, P_0 and P_1 obtain the shares of individually sorted lists of PCVs of the data owners. The proxies need to merge individually sorted lists pairwise until they obtain the global sorted list of PCVs. This can be considered as the leaves of a binary tree merging into the root node, which is, in our case, the global sorted list. Due to the high complexity of privacy preserving sorting, we decided to make the sorting parametric to adjust the trade-off between privacy and practicality. Let $\delta = 2a + 1$ be this parameter determining the number of PCVs that will be added to the global sorted list in each iteration for $a \in \mathbb{N}$, and L_{i_k} and L_{j_k} be the shares of two individually sorted

lists of PCVs in P_ks for $k \in \{0, 1\}$ and $|L_i| \geq |L_j|$ where $|.|$ is the size operator. First, the proxies privately compare the lists elementwise. They use the results of the comparison in MUXs to privately exchange the shares of PCVs in each pair, if the PCV in L_j is larger than the PCV in L_i. In the first MUX, they input the share in L_{i_k} to MUX first and then the share in L_{j_k} along with the share of the result of the comparison to select the larger of the PCVs. They move the results of the MUX to L_{i_k}. In the second MUX, they reverse the order to select the smaller of the PCVs and move it to L_{j_k}. We call this stage *shuffling*. Then, they move the top PCV of L_{i_k} to the merged list of PCVs. If $\delta \neq 1$, then they continue comparing the top PCVs in the lists and moving the largest of them to the merged list. Once they move δ PCVs to the merged list, they shuffle the lists again, and if $|L_{j_k}| > |L_{i_k}|$, then they switch the lists. Until finishing up the PCVs in L_{i_k}, the proxies follow shuffling-moving cycle.

By shuffling, we increase the number of candidates for a specific position and, naturally, lower the chance of matching a PCV in the individually sorted lists to a PCV in the merged list. The highest chance of matching is 50%, leading to a very low chance of guessing the matching of whole PCVs in the list. In sorting, δ must be an odd number to make sure that shuffling always leads to an increment in the number of candidates. An even value of δ may cause ineffective shuffling during the sorting. Although $\delta = 1$ provides the utmost privacy, which means that the chance of guessing the matching of the whole PCVs is 1 over the number of all possible merging of those two individually sorted lists, the execution time of sorting can be relatively high. For $\delta \neq 1$, the execution time can be low but the number of possible matching of PCVs in the individually sorted list to the merged list decreases in parallel to the increment of δ. As a guideline on the choice of δ, one can decide it based on how much privacy loss any matching could cause on the specific task. In case of $\delta \neq 1$ and $|L_{j_k}| = 1$ at some point in the sorting, the sorting continues as if it had just started with $\delta = 1$ to make sure that the worst case scenario for guessing the matching can be secured. More details of the sorting phase are in the Appendix.

Division (DIV): For the exact AUC, we need a division operation which is not offered by CECILIA. Therefore, we adapted the division operation from SecureNN [18]. However, we use the building blocks of CECILIA to implement the division operation since they have less communication round complexities than SecureNN. DIV uses long division to find the quotient. Although DIV of SecureNN is rather a normalization operation, requiring the denominator to be larger than the nominator, it is still useful for the exact AUC computation. In both AUROC and AUPR, the denomitors and nominators of the division operations satisfy this requirement.

4.1 Secure Computation of AUROC

Once P_0 and P_1 obtain the global sorted list of PCVs, they calculate the AUROC based on this list using one of the versions of AUROC depending on whether there exists a tie in the list.

input : $\langle L\rangle_i = (\{\langle con_1\rangle_i, \langle label_1\rangle_i\}, ..., \{\langle con_M\rangle_i, \langle label_M\rangle_i\})$, $\langle L\rangle_i$ is a share of the global sorted list of PCVs, and labels

For each $i \in \{0, 1\}$, P_i executes Steps 2-11
$\langle TP\rangle_i \leftarrow 0$, $\langle P\rangle_i \leftarrow 0$, $\langle pFP\rangle_i \leftarrow 0$, $\langle N\rangle_i \leftarrow 0$
foreach *item* $\langle t\rangle \in \langle L\rangle$ **do**
    $\langle TP\rangle_i \leftarrow \langle TP\rangle_i + \langle t.label\rangle_i$
    $\langle P\rangle_i \leftarrow \langle P\rangle_i + i$
    $\langle FP\rangle_i \leftarrow \langle P\rangle_i - \langle TP\rangle_i$
    $\langle A\rangle_i \leftarrow \mathsf{MUL}(\langle TP\rangle_i, \langle FP\rangle_i - \langle pFP\rangle_i)$
    $\langle N\rangle_i \leftarrow \langle N\rangle_i + \langle A\rangle_i$
    $\langle pFP\rangle_i \leftarrow \langle FP\rangle_i$
$\langle D\rangle_i \leftarrow \mathsf{MUL}(\langle TP\rangle_i, \langle FP\rangle_i)$
$\langle ROC\rangle_i \leftarrow \mathsf{DIV}(\langle N\rangle_i, \langle D\rangle_i)$

Algorithm 1: Secure AUROC computation without ties

input : $\langle L\rangle_i = (\{\langle con_1\rangle_i, \langle label_1\rangle_i\}, , ..., \{\langle con_M\rangle_i, \langle label_M\rangle_i\})$, $\langle L\rangle_i$ is a share of the global sorted list of PCVs, and labels

For each $i \in \{0, 1\}$, P_i executes Steps 2-14
$\langle TP\rangle_i \leftarrow 0$, $\langle P\rangle_i \leftarrow 0$, $\langle pFP\rangle_i \leftarrow 0$, $\langle pTP\rangle_i \leftarrow 0$, $\langle N_1\rangle_i \leftarrow 0$, $\langle N_2\rangle_i \leftarrow 0$
foreach *item* $\langle t\rangle_i \in \langle L\rangle_i$ **do**
    $\langle TP\rangle_i \leftarrow \langle TP\rangle_i + \langle t.label\rangle_i$
    $\langle P\rangle_i \leftarrow \langle P\rangle_i + i$
    $\langle FP\rangle_i \leftarrow \langle P\rangle_i - \langle TP\rangle_i$
    $\langle A\rangle_i \leftarrow \mathsf{MUL}([\langle pTP\rangle_i, \langle TP\rangle_i - \langle pTP\rangle_i], [\langle FP\rangle_i - \langle pFP\rangle_i, \langle FP\rangle_i - \langle pFP\rangle_i])$
    $\langle A\rangle_i \leftarrow \mathsf{MUL}(\langle A\rangle_i, [\langle t.con\rangle_i, \langle t.con\rangle_i])$
    $\langle N_1\rangle_i \leftarrow \langle N_1\rangle_i + \langle A[0]\rangle_i$
    $\langle N_2\rangle_i \leftarrow \langle N_2\rangle_i + \langle A[1]\rangle_i$
    $[\langle pre\_FP\rangle_i, \langle pre\_TP\rangle_i] \leftarrow \mathsf{MUX}([\langle pFP\rangle_i, \langle pTP\rangle_i], [\langle FP\rangle_i, \langle TP\rangle_i], [\langle t.con\rangle_i, \langle t.con\rangle_i])$
$\langle N\rangle_i \leftarrow 2 \cdot \langle N_1\rangle_i + \langle N_2\rangle_i$
$\langle D\rangle_i \leftarrow 2 \cdot \mathsf{MUL}(\langle TP\rangle_i, \langle FP\rangle_i)$
$\langle ROC\rangle_i \leftarrow \mathsf{DIV}(\langle N\rangle_i, \langle D\rangle_i)$

Algorithm 2: Secure AUROC computation with tie

Secure AUROC Computation without Ties: In Algorithm 1, we compute the AUROC as shown in Eq. 1 by assuming that there is no tie in the sorted list of PCVs. At the end of the secure computation, the shares of numerator N and denominator D are computed. Since N is always greater than or equal to D, we can use the division of SecureNN to obtain $AUROC = N/D$. With the help of high numeric value precision of the results, most of the machine learning algorithms yield different PCVs for samples. Therefore, this version of computing the AUROC is applicable to most machine learning tasks. However, in case of a tie between samples from two classes in the PCVs, it does not guarantee the exact AUROC. Depending on the order of the samples, it approximates the score. To have a more accurate AUROC, we propose another version of AUROC computation with a slightly higher communication cost in the next section.

Secure AUROC Computation with Ties: To detect ties in the list of PCVs, P_0 and P_1 compute the difference between each PCV and its following PCV. P_0 computes the modular additive inverse of its shares. The proxies apply a common random permutation to the bits of each share in the list to prevent P_2 from learning the non-zero relative differences. They also permute the list of shares using a common random permutation to shuffle the order of the real

input : $\langle C\rangle_i = (\langle con_1\rangle_i, ..., \langle con_M\rangle_i)$, $\langle C\rangle_i$ is a share of the global sorted list of PCVs, M is the number of PCVs
P_0 and P_1 hold a common random permutation π for M items
P_0 and P_1 hold a list of common random values R
P_0 and P_1 hold a list of common random permutation σ for ℓ items
For each $i \in \{0, 1\}$, P_i executes Steps 5-13
for $j \leftarrow 1$ **to** $M - 1$ **do**
  $\langle C[j]\rangle_i \leftarrow (\langle C[j]\rangle_i - \langle C[j+1]\rangle_i)$
  if $i = 0$ **then**
    $\langle C[j]\rangle_i = K - \langle C[j]\rangle_i$
  $\langle C[j]\rangle_i = \langle C[j]\rangle_i \oplus R[j]$
  $\langle C[j]\rangle_i = \sigma_j(\langle C[j]\rangle_i)$
$\langle D\rangle_i = \pi(\langle C\rangle_i)$
Insert arbitrary number of dummy zero and non-zero values to randomly chosen locations in $\langle D\rangle_i$
P_i sends $\langle D\rangle_i$ to P_2
P_2 reconstructs D by computing $\langle D\rangle_0 \oplus \langle D\rangle_1$
foreach *item* $\langle d\rangle \in \langle D\rangle$ **do**
  if $d > 0$ **then**
    $d \leftarrow 1$
P_2 creates new shares of D, denoted by $\langle D\rangle_0$ and $\langle D\rangle_1$, and sends them to P_0 and P_1, respectively.
For each $i \in \{0, 1\}$, P_i executes Steps 18-21
Remove dummy zero and non-zero values from $\langle D\rangle_i$
$\langle C\rangle_i = \pi^{-1}(\langle D\rangle_i)$
for $j \leftarrow 1$ **to** $M - 1$ **do**
  $\langle L[j].con\rangle_i \leftarrow \langle C[j]\rangle_i$
$\langle L[M].con\rangle_i \leftarrow i$

Algorithm 3: Secure detection of ties

test samples. Then, they send the list of shares to P_2. P_2 XORes two shares and maps the result to one, if it is greater than zero and zero otherwise. Then, proxies privately map PCVs to zero if they equal to their previous PCV and one otherwise. This phase is depicted in Algorithm 3. In Algorithm 2, P_0 and P_1 use these mappings to take only the PCVs which are different from their subsequent PCV into account in the computation of the AUROC based on Eq. 2. In Algorithm 2, DIV adapted from SecureNN can be used since the numerator is always less than or equal to the denominator, as in the AUROC computation.

4.2 Secure AUPR Computation

As in the AUROC with tie computation, P_0 and P_1 map a PCV in the global sorted list to 0 if it equals the previous PCV and 1 otherwise via Algorithm 3. Then, we use Eq. 2 to calculate AUPR as shown in Algorithm 4. The most significant difference of AUPR from AUROC with tie computation is that the denominator of each precision value is different in the AUPR calculation. Thus, we need to compute the precision for each iteration in advance, requiring a vectorized division operation before iterating the list of PCVs mapped to one.

input : $\langle L \rangle_i = (\{\langle con_1 \rangle_i, \langle label_1 \rangle_i\}, ..., \{\langle con_M \rangle_i, \langle label_M \rangle_i\})$, $\langle L \rangle_i$ is a share of the global sorted list of PCVs, and labels

P_0 and P_1 hold a common random permutation π for M items
For each $i \in \{0, 1\}$, P_i executes Steps 3-19
$\langle TP[0] \rangle_i \leftarrow 0$, $\langle RC[0] \rangle_i \leftarrow 0$, $\langle pPC \rangle_i \leftarrow i$, $\langle pRC \rangle_i \leftarrow 0$, $\langle N_1 \rangle_i \leftarrow 0$, $\langle N_2 \rangle_i \leftarrow 0$
for $j \leftarrow 1$ **to** M **do**
    $\langle TP[j] \rangle_i \leftarrow \langle TP[j-1] \rangle_i + \langle L[j].label \rangle_i$
    $\langle RC[j] \rangle_i \leftarrow \langle RC[j-1] \rangle_i + i$
$\langle T\_TP \rangle_i = \pi(\langle TP \rangle_i)$
$\langle T\_RC \rangle_i = \pi(\langle RC \rangle_i)$
$\langle T\_PC \rangle_i \leftarrow \mathsf{DIV}(\langle T\_TP \rangle_i, \langle T\_RC \rangle_i)$
$\langle PC \rangle_i = \pi'(\langle T\_PC \rangle_i)$
for $j \leftarrow 1$ **to** M **do**
    $\langle A \rangle_i \leftarrow \mathsf{MUL}([\langle pPC \rangle_i \langle RC[j] \rangle_i - \langle pRC \rangle_i], [\langle RC[j] \rangle_i - \langle pRC \rangle_i, \langle PC[j] \rangle_i - \langle pPC \rangle_i])$
    $\langle A \rangle_i \leftarrow \mathsf{MUL}(\langle A \rangle_i, [\langle L[j].con \rangle_i, \langle L[j].con \rangle_i])$
    $\langle N_1 \rangle_i \leftarrow \langle N_1 \rangle_i + \langle A[0] \rangle_i$
    $\langle N_2 \rangle_i \leftarrow \langle N_2 \rangle_i + \langle A[1] \rangle_i$
    $[\langle pPC \rangle_i, \langle pRC \rangle_i] \leftarrow$
    $\mathsf{MUX}([\langle pPC \rangle_i, \langle pRC \rangle_i], [\langle PC[j] \rangle_i, \langle RC[j] \rangle_i],$
    $[\langle L[j].con \rangle_i, \langle L[j].con \rangle_i])$
$\langle N \rangle_i \leftarrow 2 \cdot \langle N_1 \rangle_i + \langle N_2 \rangle_i$
$\langle D \rangle_i \leftarrow 2 \cdot \langle TP[M] \rangle_i$
$\langle PRC \rangle_i \leftarrow \mathsf{DIV}(\langle N \rangle_i, \langle D \rangle_i)$

Algorithm 4: Secure AUPR computation

5 Security Analysis

In this section, we provide semi-honest simulation-based security proofs for the computations of ppAURORA based on the security of CECILIA's building blocks.

Lemma 1. *The protocol in Algorithm 1 securely computes AUROC in the* $(\mathcal{F}_{\mathsf{MUL}}, \mathcal{F}_{\mathsf{DIV}})$ *hybrid model.*

Proof. In the protocol, we separately calculate the numerator N and the denominator D of the AUROC, which can be expressed as $AUROC = \frac{N}{D}$. Let us first focus on the computation of D. It is equal to the multiplication of the number of samples with label 1 by the number of samples with label 0. In the end, we have the number of samples with label 1 in TP and calculate the number of samples with label 0 by $P - TP$. Then, the computation of D is simply the multiplication of these two values. To compute N, we used Eq. 1. We have already shown the denominator part of it. For the numerator part, we need to multiply the current TP by the change in FP and sum up these multiplication results. $\langle A \rangle \leftarrow \mathsf{MUL}(\langle TP \rangle, \langle FP \rangle - \langle pFP \rangle)$ computes the contribution of the current sample on the denominator and we accumulate all the contributions in N, which is the numerator part of Eq. 1. Therefore, we can conclude that we correctly compute the AUROC.

Next, we prove the security of our protocol. P_i where $i \in \{0, 1\}$ sees $\{\langle A \rangle\}_{j \in M}$, $\langle D \rangle$ and $\langle ROC \rangle$, which are fresh shares of these values. Thus the view of P_i is perfectly simulatable with uniformly random values.

Lemma 2. *The protocol in Algorithm 3 securely marks the location of ties in the list of prediction confidences.*

Proof. For the correctness of our protocol, we need to prove that for each index j in L, $L[j].con = 0$ if $(C[j] - C[j+1]) = 0$, $L[j].con = 1$, otherwise. We first calculate the difference of successive items in C. Assume we have two additive shares $(\langle a\rangle_0, \langle a\rangle_1)$ of a over the ring $\mathbb{Z}_K$. If $a = 0$, then $(K - \langle a\rangle_0) \oplus \langle a\rangle_1 = 0$ and if $a \neq 0$, then $(K - \langle a\rangle_0) \oplus \langle a\rangle_1 \neq 0$ where $K - \langle a\rangle_0$ is the additive modular inverse of $\langle a\rangle_0$. We use this fact in our protocol. P_0 computes the additive inverse of each item $\langle c\rangle_0$ in $\langle C\rangle_0$ which is denoted by $\langle c\rangle_0'$, XORes $\langle c\rangle_0'$ with a common random number in R, which is denoted by $\langle c\rangle_0''$ and permutes the bits of $\langle c\rangle_0''$ with a common permutation σ which is denoted by $\langle c\rangle_0'''$. P_1 XORes each item $\langle c\rangle_1$ in $\langle C\rangle_1$ with a common random number in R which is denoted by $\langle c\rangle_1''$ and permutes the bits of $\langle c\rangle_1''$ with a common permutation σ which is denoted by $\langle c\rangle_1'''$. P_i where $i \in \{0, 1\}$ permutes values in $\langle C\rangle_i'''$ by a common random permutation π which is denoted by $\langle D\rangle_i$. After receiving $\langle D\rangle_0$ and $\langle D\rangle_1$, P_2 maps each item d of D to 0 if $\langle d\rangle_0' \oplus \langle d\rangle_1 = 0$ which means $\langle d\rangle_0 + \langle d\rangle_1 = 0$ and maps 1 if $\langle d\rangle_0' \oplus \langle d\rangle_1 \neq 0$ which means $\langle d\rangle_0 + \langle d\rangle_1 \neq 0$. After receiving a new share of D from P_2, P_i where $i \in \{0, 1\}$ removes dummy values and permutes remaining values by π^{-1}. Therefore, our protocol correctly maps items of C to 0 or 1.

We next prove the security of our protocol. P_i where $i \in \{0, 1\}$ calculates the difference of successive prediction values. The view of P_2 is D, which includes real and dummy zero values. P_i XORes each item of $\langle C\rangle_i$ with fresh boolean shares of zero, applies a random permutation to bits of each item of $\langle C\rangle_i$, applies a random permutation π to $\langle C\rangle_i$ and add dummy zero and non-zero values. Thus the differences, the positions of the differences, and the distribution of the differences are completely random. The number of zero and non-zero values are not known to P_2 due to dummy values. With common random permutations $\sigma_{j\in M}$ and common random values $R[j], j \in M$, each item in C is hidden. Thus P_2 can not infer anything about real values in C. Furthermore, the number of repeating predictions is not known to P_2 due to the random permutation π.

Lemma 3. *The protocol in Algorithm 2 securely computes AUROC in ($\mathcal{F}_{\mathsf{MUL}}$, $\mathcal{F}_{\mathsf{MUX}}$, $\mathcal{F}_{\mathsf{DIV}}$) hybrid model.*

Proof. To compute the AUROC in case of a tie, we use Eq. 2, of which we calculate the numerator and the denominator separately. The calculation of the denominator D is the same as Lemma 1. The computation of the numerator N has two different components, which are N_1 and N_2. N_1, more precisely the numerator of $T[i-1] * (F[i] - F[i-1])$, is similar to the no-tie version of privacy preserving AUROC computation. This part corresponds to the rectangle areas in the ROC curve. The decision of adding this area A to the cumulative area N_1 is made based on the result of the multiplication of A by $L.con$. $L.con = 1$ indicates if the sample is one of the points of prediction confidence change, 0 otherwise. If it is 0, then A becomes 0 and there is no contribution to N_1. If it is 1, then we add A to N_1. On the other hand, N_2, which is the numerator of

$(T[i]-T[i-1])*(F[i]-F[i-1])$, accumulates the triangular areas. We compute the possible contribution of the current sample to N_2. In case this sample is not one of the points that the prediction confidence changes, which is determined by $L.con$, then the value of A is set to 0. If it is, then A remains the same. Finally, A is added to N_2. Since there is a division by 2 in the second part of Eq. 2, we multiply N_1 by 2 to make them have common denominator. Then, we sum N_1 and N_2 to obtain N. To have the term 2 in the common denominator, we multiply D by 2. As a result, we correctly compute the denominator and the nominator of the AUROC.

Next, we prove the security of our protocol. P_i where $i \in \{0,1\}$ sees $\{\langle A\rangle\}_{j\in M}$, $\{\langle pFP\rangle\}_{j\in M}$, $\{\langle pTP\rangle\}_{j\in M}$, $\langle D\rangle$ and $\langle ROC\rangle$, which are fresh shares of them. Thus the view of P_i is perfectly simulatable with uniformly random values.

Lemma 4. *The protocol in Algorithm 4 securely computes AUPR in ($\mathcal{F}_{\mathsf{MUL}}$, $\mathcal{F}_{\mathsf{MUX}}$, $\mathcal{F}_{\mathsf{DIV}}$) hybrid model.*

Proof. To compute the AUPR, we use Eq. 2 of which we calculate the numerator and the denominator separately. We nearly perform the same computation with the AUROC with tie computation. The main difference is that we need to perform a division to calculate each precision value because denominators of each precision value are different. The rest of the computation is the same with the computation in Algorithm 2. The readers can follow the proof of Lemma 3.

Next, we prove the security of our protocol. P_i where $i \in \{0,1\}$ sees $\{\langle T\_PC\rangle\}_{j\in M}$, $\{\langle A\rangle\}_{j\in M}$, $\{\langle pPC\rangle\}_{j\in M}$, $\{\langle pRC\rangle\}_{j\in M}$ and $\langle PRC\rangle$, which are fresh shares of them. Thus the view of P_i is perfectly simulatable with uniformly random values.

Lemma 5. *The sorting protocol in Sect. 4 securely merges two sorted lists in ($\mathcal{F}_{\mathsf{CMP}}$,$\mathcal{F}_{\mathsf{MUX}}$) hybrid model.*

Proof. First, we prove the correctness of our merge sorting of lists L_1 and L_2. In the merging of L_1 and L_2, the corresponding values are first compared using CMP operation. The larger values are placed in L_1 and the smaller values are placed in L_2, after MUX operation is called twice. This process is called *shuffling* because it shuffles the corresponding values in the two lists. After the shuffling, we know that the largest element of the two lists is the top element of L_1. Thus, it is moved to the global sorted list L_3. On the next step, the top elements of L_1 and L_2 are compared with CMP method. The comparison result is reconstructed by P_0 and P_1 and the top element of L_1 or L_2 is moved to L_3 based on the result of CMP. The selection operation also gives the largest element of L_1 and L_2 because L_1 and L_2 are sorted. We show that shuffling and selection operations give the largest element of two sorted lists. This ensures that our merge sort algorithm that only uses these operations correctly merges two sorted lists privately.

Next, we prove the security of our merge sort algorithm in which CMP and MUX are called. CMP outputs fresh shares of comparison of corresponding values in L_1 and L_2. Shares of these comparison results are used in MUX that generates fresh shares of the corresponding values. Thus, P_0 and P_1 cannot precisely map

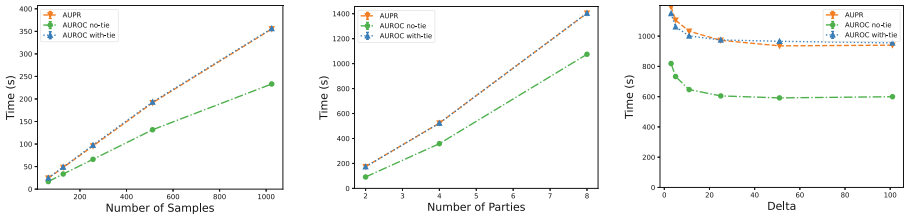

Fig. 2. The scalability of ppAURORA to varying **(a)** number of samples, **(b)** number of parties and **(c)** δ where the other parameters are fixed

these values to the values in L_1 and L_2. In the selection operation, CMP is called and its reconstructed output is used to select. P_0 and P_1 are still unable to map the values added to L_3 to the values in L_1 and L_2 precisely since at least one shuffling operation took place before these repeated selection operations. Shuffling and $\delta - 1$ selection operations are performed repeatedly until the L_1 is empty. After each shuffling operation, the fresh share of the larger corresponding values in L_1 and the fresh share of the smaller corresponding values in L_2 are stored. The view of P_0 and P_1 are perfectly simulatable with random values due to the shuffling process performed at regular intervals.

To prevent the usage of unshuffled values in some cases, the following rules are followed in the execution of the merge protocol. If there are two lists that do not have the same length, the longer list is chosen as L_1. If the δ is greater than the length of the L_2 list, it is set to the largest odd value smaller or equal to the length of L_2 so that the unshuffled values that L_1 may have are not used in selection processes. If the length of L_2 is reduced to 1 at some point in the sorting, the δ is set to 1. Thus L_2 will have 1 element until the end of the merge and shuffling is done before each selection. After moving δ values to the sorted list, if the length of L_2 is greater than the length of L_1, we switch the list.

5.1 Privacy Against Malicious Adversaries

Araki et al. [1] defined a privacy notion against malicious adversaries in the client-server setting where the servers performing secure computation on the shares of the inputs to produce the shares of the outputs do not see the plain inputs and outputs of the clients, which is very similar to our setting. In our framework, two parties exchange a seed to generate common random values between them. Two parties randomize their shares using these random values, which are unknown to the third party. It is very easy to add fresh shares of zero to outputs of two parties with common random values shared between them. In our algorithms, we do not state the randomization of outputs with fresh shares of zero. Thus, our framework provides privacy against a malicious party by relying on the security of a seed shared between two honest parties.

6 Results

Dataset: We used the Acute Myeloid Leukemia (AML) dataset[1] and the UCI Heart Disease dataset[2] for the correctness analysis of ppAURORA. AML dataset is from the submission of the team *Snail*, which has the lowest score, in the first subchallenge of the DREAM Challenge [14] and has 191 samples, among which 136 patients have complete remission. UCI Heart Disease test set has 54 samples with binary outcome. Moreover, we aimed to analyze the scalability of ppAURORA for different settings. For this purpose, we generated a synthetic dataset with no restriction other than having the PCVs from $[0, 1]$.

Experimental Setup: We conducted our experiments on LAN and WAN settings. In the LAN, we ran the experiments with 0.18 ms round trip time (RTT). In the WAN, we simulated the network connection with 10 ms RTT.

Correctness Analysis: We conducted the correctness analysis on the LAN setting. To assess the correctness of AUROC with tie, we computed the AUROC of the predictions on the AML dataset by ppAURORA, yielding $AUROC = 0.693$ which is the same the result obtained without privacy on the DREAM Challenge dataset. For the correctness of AUROC with no-tie of ppAURORA, we randomly picked one of the samples in tie condition in DREAM Challenge dataset and generated a subset of the samples with no tie. We got the same AUROC with no-tie version of AUROC of ppAURORA as the non-private computation. We directly used the UCI dataset in AUROC with no-tie since it does not have any tie condition. The result, which is $AUROC = 0.927$, is the same for both private and non-private computation. Besides, we verified that ppAURORA computes the same AUPR as for the non-private computation for both the DREAM Challenge and the UCI dataset, which are $AUPR = 0.844$ and $AUPR = 0.893$, respectively. These results indicate that ppAURORA can privately compute the exact same AUC as one could obtain on the pooled test samples.

Scalability Analysis: We evaluated the scalability of no-tie and with-tie versions of AUROC and AUPR of ppAURORA to the number of samples $M \in \{64, 128, 256, 512, 1024\}$ with $\delta = 1$ and 3 data sources. The results showed that ppAURORA scales almost quadratically in terms of both communication costs among all parties and the execution time of the computation. We also analyzed the performance of all computations of ppAURORA on a varying number of data sources. We fixed $\delta = 1$ and the number of samples in each data sources to 1000, and we experimented with D data sources where $D \in \{2, 4, 8\}$. ppAURORA scales around quadratically to the number of data sources. We also analyzed the effect of $\delta \in \{3, 5, 11, 25, 51, 101\}$ by fixing D to 8 and M in each data source to 1000. The execution time displays a logarithmic decrease for increasing δ. In all analyses, since the dominating factor is sorting, the execution times of the computations are close to each other. Additionally, our analysis showed that LAN is 12 to 14 times faster than WAN on average due to the high

[1] https://www.synapse.org/#!Synapse:syn2700200.
[2] https://archive.ics.uci.edu/ml/datasets/heart+disease.

Table 1. The results of AUPR computation with ppAURORA where D is the number of data sources and M is the number of samples in one data source. UNB, i.e. unbalanced sample distribution, is $\{12, 18, 32, 58, 107, 258, 507, 1008\}$.

D × M	δ	Communication Costs (MB)				Time (sec)
		P_1	P_2	Helper	Total	
3×64	1	1.96	1.3	1.13	4.39	24.41
3×128	1	6.61	4.14	3.97	14.72	48.05
3×256	1	24.44	15.23	15.06	54.73	95.65
3×512	1	93.23	58.44	58.26	209.93	191.55
3×1024	1	359.67	226.62	226.41	812.7	355.32
2×1000	1	125.05	78.37	78.19	281.61	174.16
4×1000	1	726.44	458.81	458.57	1643.82	523.39
8×1000	1	3355.74	2125.91	2125.51	7607.16	1404.22
8×1000	3	1692.85	1069.58	1069.25	3831.68	1194.08
8×1000	5	1137.91	717.45	717.15	2572.51	1105.46
8×1000	11	583.02	365.36	365.08	1313.46	1032.29
8×1000	25	284.22	175.76	175.5	635.48	972.0
8×1000	51	156.23	94.54	94.29	345.06	935.65
8×1000	101	93.59	54.79	54.53	202.91	940.07
$8 \times UNB$	1	130.48	81.82	81.6	293.9	379.99

round trip time of WAN, which is approximately 10 ms. Even with such a scaling factor, ppAURORA can be deployed in real life scenarios if the alternative is a more time-consuming approval process required for gathering all data in one place still protecting the privacy of data. Figure 2 and Table 1 display the results.

7 Conclusion

In this work, we presented an efficient and exact solution based on a secure 3-party computation framework to compute AUC of the ROC and PR curves privately even when there exist ties in the PCVs. We benefited from the built-in building blocks of CECILIA and adapted the division operation of SecureNN to compute the exact AUC. ppAURORA is secure against passive adversaries in the honest majority setting. We demonstrated that ppAURORA can compute correctly and privately the exact AUC that one could obtain on the pooled plaintext test samples, and ppAURORA scales quadratically to the number of

both parties and samples. In future work, we will further optimize the sorting phase in terms of both privacy and efficiency.

Acknowledgement. This study is supported by the DFG Cluster of Excellence "Machine Learning - New Perspectives for Science", EXC 2064/1, project number 390727645 and the German Ministry of Research and Education (BMBF), project number 01ZZ2010.

References

1. Araki, T., Furukawa, J., Lindell, Y., Nof, A., Ohara, K.: High-throughput semi-honest secure three-party computation with an honest majority. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 805–817 (2016)
2. Boyd, K., Lantz, E., Page, D.: Differential privacy for classifier evaluation. In: Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security, pp. 15–23 (2015)
3. Byali, M., Chaudhari, H., Patra, A., Suresh, A.: Flash: fast and robust framework for privacy-preserving machine learning. Proc. Priv. Enh. Technol. **2020**(2), 459–480 (2020)
4. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14–17 October 2001, Las Vegas, Nevada, USA, pp. 136–145. IEEE Computer Society (2001). https://doi.org/10.1109/SFCS.2001.959888
5. Chaudhuri, K., Vinterbo, S.A.: A stability-based validation procedure for differentially private machine learning. In: Advances in Neural Information Processing Systems, pp. 2652–2660 (2013)
6. Chen, Y., Machanavajjhala, A., Reiter, J.P., Barrientos, A.F.: Differentially private regression diagnostics. In: ICDM, pp. 81–90 (2016)
7. Damgård, I., Escudero, D., Frederiksen, T., Keller, M., Scholl, P., Volgushev, N.: New primitives for actively-secure MPC over rings with applications to private machine learning. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 1102–1120. IEEE (2019)
8. Juvekar, C., Vaikuntanathan, V., Chandrakasan, A.: {GAZELLE}: a low latency framework for secure neural network inference. In: 27th {USENIX} Security Symposium ({USENIX} Security 2018), pp. 1651–1669 (2018)
9. Li, B., Wu, Y., Song, J., Lu, R., Li, T., Zhao, L.: Deepfed: federated deep learning for intrusion detection in industrial cyber-physical systems. IEEE Trans. Industr. Inf. **17**(8), 5615–5624 (2020)
10. Lindell, Y.: How to simulate it – a tutorial on the simulation proof technique. In: Lindell, Y. (ed.) Tutorials on the Foundations of Cryptography. ISC, pp. 277–346. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57048-8_6
11. Matthews, G.J., Harel, O.: An examination of data confidentiality and disclosure issues related to publication of empirical ROC curves. Acad. Radiol. **20**(7), 889–896 (2013)
12. Mohassel, P., Rindal, P.: ABY3: a mixed protocol framework for machine learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 35–52 (2018)

13. Mohassel, P., Zhang, Y.: Secureml: a system for scalable privacy-preserving machine learning. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 19–38. IEEE (2017)
14. Noren, D.P., et al.: A crowdsourcing approach to developing and assessing prediction algorithms for AML prognosis. PLoS Comput. Biol. **12**(6), e1004890 (2016)
15. Patra, A., Suresh, A.: BLAZE: blazing fast privacy-preserving machine learning. In: 27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, 23–26 February 2020. The Internet Society (2020)
16. Sun, J., Yang, X., Yao, Y., Xie, J., Wu, D., Wang, C.: Differentially private AUC computation in vertical federated learning. arXiv preprint arXiv:2205.12412 (2022)
17. Ünal, A.B., Akgün, M., Pfeifer, N.: CECILIA: comprehensive secure machine learning framework. CoRR abs/2202.03023 (2022). https://arxiv.org/abs/2202.03023
18. Wagh, S., Gupta, D., Chandran, N.: SecureNN: efficient and private neural network training. IACR Cryptology ePrint Archive, vol. 2018, p. 442 (2018)
19. Whitehill, J.: How does knowledge of the AUC constrain the set of possible ground-truth labelings? In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 5425–5432 (2019)