Towards a Complete Privacy Preserving Machine Learning Pipeline

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
M. Sc. Ali Burak Ünal
aus Karaman, Türkei

Tübingen
2022

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	29.07.2022
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Nico Pfeifer
2. Berichterstatter:	Prof. Dr. Sven Nahnsen

Acknowledgments

First of all, I would like to express my gratitude to my supervisor, Prof. Dr. Nico Pfeifer. I deeply thank him for his help, understanding, support and kindness throughout my doctoral study. I was very fortunate to be a member of his Methods in Medical Informatics group. I was also very fortunate to work with Dr. Mete Akgün. I am truly grateful for the countless and fruitful discussions we had. I have learned a lot from him. I would also like to thank Prof. Dr. Sven Nahnsen, Prof. Dr. Enkelejda Kasneci and Prof. Dr. Michael Menth for the valuable time they spent evaluating my dissertation.

I would like to thank Dr. Efe Bozkır, Huajie Chen and again Prof. Dr. Enkelejda Kasneci for their valuable collaboration. It was a great experience to work with them.

I am thankful for all my colleagues. It was so great to discuss science, to have small talks, to have lunch together in the offices and at Unckel, but most importantly to eat cake together at every possible occasion. Moreover, it was so nice to share the same office with Lisa Eisenberg and Nurhan Arslan. I will definitely miss those times. And a special thanks to Agnes Molden for her help with almost all problems I encountered.

Life in Tübingen would have been a bit boring without my friends here. I am so glad to have them. We have had so much fun together and played countless but really countless games. I am thankful for all of them.

Besides my friends here, I am also very grateful to my friends from Bilkent University and even from earlier stages of my life. I want to thank Emre Kahrıman, Kenan Sevinç, Onur Konukcu, Şaban Okka, Alper Şener, Raşit Somuncu, Kamil Yaprakcı and Hüseyin Açacak. These are just to name a few. Also, I cannot express how precious Ahmet Küçük's friendship is for me. It was so great to talk to him about everything until late at night and come to no conclusion in the end. His support means a lot.

I am also always grateful to Emin Yıldırım, Onur Barut and Hüseyin Kılıç. I cannot even express how valuable they are to me.

Most of all, I would like to express my gratitude to my mother Ayşe, my father İsmet, my brother Çağrı and my sister Duygu. Their love and support mean so much to me. And finally, I am so happy to share my life with my wife Elif. I cannot express how much you supported me during my doctoral study. I do not know how I would complete my Ph.D. without your support and love.

Oh, the birds of Tübingen... Thanks a lot guys for your modelling.

Ali Burak Ünal

Abstract

Machine learning has proven its success on various problems from many different domains. Different machine learning algorithms use different approaches to capture the underlying patterns in the data. Even though the amount varies between the machine learning algorithms, they require sufficient amounts of data to recognize those patterns. One of the easiest ways to meet this need of the machine learning algorithms is to use multiple sources generating the same type of data. Such a solution is feasible considering that the speed of data generation and the number of sources generating these data have been increasing in parallel to the developments in technology. One can easily satisfy the desire of the machine learning algorithms for data using these sources. However, this can cause a privacy leakage. The data generated by these sources may contain sensitive information that can be used for undesirable purposes. Therefore, although the machine learning algorithms demand for data, the sources may not be willing or even allowed to share their data. A similar dilemma occurs when the data owner wants to extract useful information from the data by using machine learning algorithms but it does not have enough computational power or knowledge. In this case, the data source may want to outsource this task to external parties that offer machine learning algorithms as a service. Similarly, in this case, the sensitive information in the data can be the decisive factor for the owner not to choose outsourcing, which then ends up with non-utilized data for the owner. In order to address these kinds of dilemmas and issues, this thesis aims to come up with a complete privacy preserving machine learning pipeline. It introduces several studies that address different phases of the pipeline so that all phases of a machine learning algorithm can be performed privately. One of these phases addressed in this thesis is training of a machine learning algorithm. The privacy preserving training of kernel-based machine learning algorithms are addressed in several different works with different cryptographic techniques, one of which is a our newly developed encryption scheme. The different techniques have different advantages over the others. Furthermore, this thesis introduces our study addressing the testing phase of not only the kernel-based machine learning algorithms but also a special type of recurrent neural network, namely recurrent kernel networks, which is the first study performing such an inference, without compromising privacy. To enable the privacy preserving inference on recurrent kernel networks, this thesis introduces a framework, called CECILIA, with two novel functions, which are the exponential and the inverse square root of the Gram matrix, and efficient versions of the existing functions, which are the multiplexer and the most significant bit. Using this framework and other approaches in the corresponding studies, it is possible to perform privacy preserving inference on various pre-trained machine learning algorithms. Besides the training and testing of machine learning algorithms in a privacy preserving way, this thesis also presents a work that aims to evaluate the performance of machine learning algorithms without sacrificing privacy. This work employs CECILIA to realize the area under curve calculation for two different curve-based evaluations, namely the receiver operating characteristic curve and the precision-recall curve, in a privacy preserving manner. All the proposed approaches are shown to be correct using several machine learning tasks and evaluated for the scalability of the parameters of the corresponding system/algorithm using synthetic data. The results show that the privacy preserving training and testing of kernel-based machine learning algorithms is possible with different settings and the privacy preserving inference on a pre-trained recurrent kernel network is feasible using CECILIA. Additionally, CECILIA also allows the exact area under curve computation to evaluate the performance of a machine learning algorithm without compromising privacy.

Zusammenfassung

Das maschinelle Lernen hat seinen Erfolg bei verschiedenen Problemen in vielen unterschiedlichen Bereichen bewiesen. Verschiedene Algorithmen für maschinelles Lernen verwenden unterschiedliche Ansätze, um die zugrunde liegenden Muster in den Daten zu erfassen. Auch wenn die Menge der Daten bei den verschiedenen Algorithmen für maschinelles Lernen unterschiedlich ist, benötigen sie doch eine ausreichende Menge an Daten, um diese Muster zu erkennen. Eine der einfachsten Möglichkeiten, diesen Bedarf der Algorithmen für maschinelles Lernen zu decken, ist die Verwendung mehrerer Quellen, die die gleiche Art von Daten erzeugen. Eine solche Lösung ist machbar, wenn man bedenkt, dass die Geschwindigkeit der Datengenerierung und die Anzahl der Quellen, die diese Daten generieren, parallel zu den Entwicklungen in der Technologie gestiegen sind. Der Wunsch der Algorithmen des maschinellen Lernens nach Daten kann mit Hilfe dieser Quellen leicht erfüllt werden. Dies kann jedoch zu einer Beeinträchtigung der Privatsphäre führen. Die von diesen Quellen erzeugten Daten können sensible Informationen enthalten, die für unerwünschte Zwecke verwendet werden können. Obwohl die Algorithmen für maschinelles Lernen Daten benötigen, sind die Quellen daher möglicherweise nicht bereit, ihre Daten weiterzugeben. Ein ähnliches Dilemma tritt auf, wenn der/die Dateneigentümer*in mit Hilfe von Algorithmen für maschinelles Lernen nützliche Informationen aus den Daten extrahieren möchte, aber nicht über genügend Rechenleistung oder Wissen verfügt. In diesem Fall kann diese Aufgabe möglicherweise an externe Parteien ausgelagert werden, die Algorithmen für maschinelles Lernen als Dienstleistung anbieten. Auch in diesem Fall können die sensiblen Informationen in den Daten der entscheidende Faktor für den/die Eigentümer*in sein, sich nicht für eine Auslagerung zu entscheiden, was dann dazu führt, dass die Daten für den/die Eigentümer*in nicht genutzt werden. Um diese Art von Dilemmata und Problemen anzugehen, zielt diese Arbeit darauf ab, eine vollständige Pipeline für maschinelles Lernen unter Wahrung der Privatsphäre zu entwickeln. Es werden mehrere Studien vorgestellt, die sich mit verschiedenen Phasen der Pipeline befassen, so dass alle Phasen eines Algorithmus für maschinelles Lernen unter Wahrung der Privatsphäre durchgeführt werden können. Eine dieser Phasen, die in dieser Arbeit behandelt wird, ist das Training eines maschinellen Lernalgorithmus. Das Training von kernbasierten maschinellen Lernalgorithmen unter Wahrung der Privatsphäre wird in verschiedenen Arbeiten mit unterschiedlichen kryptographischen Techniken behandelt, von denen eine ein von aus entwickeltes neuartiges Verschlüsselungsverfahren ist. Diese haben jeweils unterschiedliche Vorteile gegenüber den anderen. Darüber hinaus werden in dieser Arbeit Studien vorgestellt, die sich mit der Testphase nicht nur kernelbasierter maschineller Lernalgorithmen befassen, sondern auch mit einem speziellen Typ rekurrenter neuronaler Netze, nämlich den rekurrenten Kernnetzen, das die erste Studie ist, die eine solche Inferenz durchführt, ohne die Privatsphäre zu gefährden. Um eine datenschutzkonforme Inferenz auf rekurrenten Kernnetzen zu ermöglichen, wird in dieser Arbeit ein Framework mit dem Namen CECILIA eingeführt, das zwei neuartige Funktionen enthält, nämlich die Exponentialfunktion und die inverse Quadratwurzel der Gram-Matrix, sowie effiziente Versionen etablierter Funktionen, Multiplexer und least significant bit. Unter Verwendung dieses Frameworks und anderer Ansätze in den entsprechenden Studien ist es möglich, datenschutzkonforme Inferenzen für verschiedene vortrainierte Algorithmen des maschinellen Lernens durchzuführen. Neben dem Training und Testen von maschinellen Lernalgorithmen unter Wahrung der Privatsphäre wird in dieser Arbeit auch eine Studie vorgestellt, die darauf abzielt, die Leistung von maschinellen Lernalgorithmen zu bewerten, ohne die Privatsphäre zu gefährden. In dieser Arbeit wird CECILIA eingesetzt, um die Berechnung der Fläche unter der Kurve für zwei verschiedene kurrenbasierte Auswertungen, nämlich die Receiver-Operating-Characteristic-Kurve und die Precision-Recall-Kurve, auf eine datenschutzfreundliche Weise zu realisieren. Alle vorgeschlagenen Ansätze werden anhand verschiedener Aufgaben des maschinellen Lernens auf ihre Korrektheit geprüft und auf ihre Skalierbarkeit mit den Parametern des entsprechenden Systems/Algorithmus unter Verwendung synthetischer Daten untersucht. Die Ergebnisse zeigen, dass das Training und Testen von kernbasierten maschinellen Lernalgorithmen unter Wahrung der Privatsphäre mit verschiedenen Einstellungen möglich ist und, dass die Inferenz mit einem vortrainierten rekurrenten Kernnetzwerk unter Verwendung von CECILIA möglich ist. Darüber hinaus ermöglicht CECILIA auch die exakte Berechnung der Fläche unter der Kurve, um die Leistung eines maschinellen Lernalgorithmus zu bewerten, ohne die Privatsphäre zu beeinträchtigen.

Contents

List of Figures

List of Tables

Acronyms

AUC	Area Under Curve
AUPR	Area Under Precision Recall Curve
AUROC	Area Under Receiver Operating Characteristic Curve
CNN	Convolutional Neural Networks
DNN	Deep Neural Network
DP	Differential Privacy
FP	False Positive
FPR	False Positive Rate
FHE	Fully Homomorphic Encryption
GDPR	General Data Protection Regulation
HE	Homomorphic Encryption
ML	Machine Learning
MPC	Multi-party Computation
PR	Precision Recall
PCV	Prediction Confidence Value
RBF	Radial Basis Function
RE	Randomized Encoding
ReLU	Rectified Linear Unit
RKN	Recurrent Kernel Network
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
SVR	Support Vector Regression
TP	True Positive
TPR	True Positive Rate
VR	Virtual Reality

1 List of Publications

Accepted Publications

1. **Ali Burak Ünal**, Mete Akgün, and Nico Pfeifer. “A framework with randomized encoding for a fast privacy preserving calculation of non-linear kernels for machine learning applications in precision medicine.” International Conference on Cryptology and Network Security. Springer, Cham, 2019.
2. Efe Bozkir*, **Ali Burak Ünal***, Mete Akgün, Enkelejda Kasneci, and Nico Pfeifer. “Privacy preserving gaze estimation using synthetic images via a randomized encoding based framework.” ACM Symposium on Eye Tracking Research and Applications. 2020.
3. **Ali Burak Ünal**, Mete Akgün, and Nico Pfeifer. "ESCAPED: Efficient Secure and Private Dot Product Framework for Kernel-based Machine Learning Algorithms with Applications in Healthcare." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 35. No. 11. 2021.
4. Huajie Chen, **Ali Burak Ünal**, Mete Akgün, and Nico Pfeifer.“Privacy-preserving SVM on outsourced genomic data via secure multi-party computation.” Proceedings of the Sixth International Workshop on Security and Privacy Analytics. 2020.

Submitted Manuscripts

1. **Ali Burak Ünal**, Mete Akgün, and Nico Pfeifer. “CECILIA: Comprehensive Secure Machine Learning Framework” arXiv preprint arXiv:2202.03023 (2022).
2. **Ali Burak Ünal**, Mete Akgün, and Nico Pfeifer. “ppAURORA: Privacy Preserving Area Under Receiver Operating Characteristic and Precision-Recall Curves with Secure 3-Party Computation.” arXiv preprint arXiv:2102.08788 (2021).

* indicates equal contribution

1.1 Scientific Contribution

This thesis advances the literature of privacy preserving machine learning by introducing novel crytographic techniques that are used to make not well-studied machine learning algorithms private and making untouched machine learning algorithms and evaluation metrics privacy preserving via novel, effective and private functions based on well-known crytographic techniques. Towards achieving the goal of this thesis, which is a complete privacy preserving pipeline, Chapter 2 introduces the preliminary concepts and the existing approaches in the literature. Chapter 3 gives the objectives and the expected outcomes of the thesis in more detail. Following the objectives, Chapter 4 summarizes six scientific publications that form the basis of this thesis by stating their motivations, the work done, the results of the corresponding study and the discussion of these results in terms of their contribution to the literature and towards the objectives of this thesis. Chapter 5 wraps up the thesis by summarizing the discussions of the publications.

Personal Contributions

In the publication 1, the data preprocesssing and preparation was done by ABÜ. The method development was done by ABÜ with the inputs from MA and NP. ABÜ implemented the method and conducted the experiments with discussing the setup with MA and NP. The analysis was mainly done by ABÜ with the help of MA and NP. ABÜ wrote the most part of the manuscript by taking the feedback of MA and NP into account.

Regarding the publication 2, EB performed the data generation and preparation with inputs from ABÜ. Improvement on the existing framework was done by ABÜ and MA with inputs from EB. Conducting the experiments were done by ABÜ with the help of EB. ABÜ and EB performed the analysis of the results with the help of MA, EK and NP. The manuscript was written by ABÜ and EB by taking the feedback of the other authors into account.

In the publication 3, ABÜ did the data preprocessing and preparation for the experiments. In the development of the methods utilized in the study, ABÜ took a leading part with the insights of MA and NP. The implementation of the developed methods was done by ABÜ with the help of MA. ABÜ conducted the experiments on the data to validate the correctness of the method and analyze the scalability of the method. The results of these experiments were analyzed by ABÜ with inputs from MA and NP. ABÜ was responsible for the most of the manuscript and wrote it with the help of MA and NP.

In the publication 4, ABÜ and HC processed and prepared the data for the experiments. HC developed and implemented the method with inputs from MA and ABÜ. The experiments performed were conducted by HC with the help of MA and ABÜ. ABÜ, HC and MA analyzed the results of the experiments with inputs from NP. Most of the manuscript was written by HC with the help of ABÜ and MA. NP revised the manuscript at the end.

ABÜ did the preprocessing and the preparation of the data utilized in the submitted manuscript 1. ABÜ developed the methods with the help of MA and NP. The implementation of the methods was done by ABÜ with inputs from MA. ABÜ conducted the experiments to show the correctness and the scalability of the developed methods with inputs from MA and NP. ABÜ analyzed the results by considering the feedback of MA and NP. Almost all parts of the manuscript was written by ABÜ with the help of MA and NP.

The data preprocessing and preparation for the submitted manuscript 2 were done by ABÜ with inputs from MA. MA and ABÜ developed the methods employed in the publication. The implementation of the methods was done by MA with inputs from ABÜ. MA conducted the experiments with the help of ABÜ. The analysis of the results was done by ABÜ with inputs from MA and NP. ABÜ wrote the most part of the manuscript with inputs from MA and NP.

2 Introduction

Data generation has increased at a tremendous rate in today's world. In 2020, a single person generates an average of 1.7 MB of data per second [3]. The sources for this big data are also increasing. They range from social media to mapping applications, from health data to email, from countless images to voice recordings. All of this data contains specific information about the owner and can be used to improve a person's quality of life through better diagnosis of diseases, personalized recommendations, autocomplete typing, and so on. Even a person can have their DNA sequenced at an affordable price to learn what genetic diseases they might get and take precautions in advance to prevent them [4]. However, this data can also be used against these individuals. The sensitive information in such personal data can threaten not only the privacy of the individual, but also that of their loved ones [5, 6, 7]. Therefore, it is extremely important to protect the data that contains sensitive information.

On the one hand, data privacy must be preserved. On the other hand, data processing approaches must be used to extract useful information. One of these approaches is machine learning (ML) algorithms. They have become increasingly successful. There are countless studies in the literature using ML algorithms to successfully address a wide range of problems that have been previously unsolved or poorly solved [8, 9, 10, 11, 12, 13]. The algorithms used range from traditional machine learning algorithms such as support vector machine (SVM), logistic regression, random forests, linear regression, etc., to more complex deep neural networks (DNNs), which have become popular and demonstrably successful at certain tasks in parallel with the increase in computer processing power.

Although the amount varies among the ML algorithms, one of the common features of these algorithms is that they require enough data to perform well. One can improve the performance of the chosen ML algorithm significantly by increasing the number of data samples that are used in the training phase of the algorithm. In particular, for DNNs, a large set of samples with high coverage is one of the crucial criteria for high performance. The demand of DNNs for data increases in parallel with the increase in the complexity of its network architecture. The simplest way to satisfy the ML algorithms' need for data is to use multiple data sources that produce the same type of data. With this approach, one can train an ML algorithm by benefiting from the data of multiple sources whose data alone may not be sufficient for training.

However, such an approach to solving the problem of having enough data risks exposing the privacy of the data used to train an ML algorithm. Releasing the data with sensitive information also reveals the sensitive information about the owner of the data. To prevent such data leaks, there are various regulations such as the General Data Protection Regulation (GDPR) [14]. Such regulations legally aim to protect the privacy of data owners by prohibiting the release and use of data in certain ways. This conflict of interest between the privacy of data and the demand of ML algorithms for data creates a dilemma and forces researchers to develop new approaches to meet the needs of both sides. These efforts have led to the development of privacy preserving ML algorithms.

This thesis advances the field of privacy preserving ML by introducing both novel cryptographic techniques and novel and efficient functions based on well-known cryptographic techniques to enable privacy preserving training and testing of several important ML algorithms, some of which have not been addressed before. It also presents the applications of these privacy preserving ML algorithms to problems in healthcare, computational biology, and human-computer interfaces to demonstrate the feasibility of performing these tasks privately by exploiting these novel approaches. Later in the Introduction, Section 2.1 introduces the ML algorithms that this thesis aims to make private, and Section 2.3 gives a brief overview of the cryptographic techniques used to make these algorithms privacy preserving. Finally, Section 2.4 presents the existing approaches in the literature that propose different privacy preserving ML algorithms using different cryptographic techniques.

2.1 Targeted Machine Learning Algorithms

This section presents the ML algorithms that are subject of this thesis before the details of how these algorithms are made private.

2.1.1 Oligo Kernel

String kernels can be thought of a way to represent similarity of sequences enabling a set of ML algorithms such as SVMs and Gaussian processes. They aim to measure the similarities between strings with different mechanisms and, based on the resulting similarity matrix, perform the intended task, which can be classification, clustering or regression, based on the resulting similarity matrix. Considering the different types of genomic sequences, they are of great importance in the medical domain to measure and/or compare these sequences leading to different applications and fields such as comparative genomics, stratification of cancer patients into subgroups and so on. One of these string kernels is the oligo kernel [15]. It measures the similarity between two sequences based on the occurrences of *oligomers*, which are basically k-mers, by taking into account the positional variations of the oligomers. Let x and y be the sequences over an alphabet $\mathscr{A}$. The similarity between sequences x and y is measured as follows:

$$K(x,y) = \sqrt{\pi}\sigma \sum_{\omega \in \mathscr{A}^k} \sum_{p \in x_\omega} \sum_{q \in y_\omega} exp(-\frac{1}{4\sigma^2}(p-q)^2) \tag{2.1}$$

where $\mathscr{A}^k$ contains all possible oligomers of length k over the alphabet $\mathscr{A}$, x_ω and y_ω are the set of positions of all occurrences of the oligomer ω in the sequences x and y, respectively, and σ is the positional uncertainty parameter that adjusts the contribution of oligomers with different positions to the similarity score. When σ is close to 0, only the oligomers that occur at the same position in both sequences contribute to the similarity score of the sequences. On the other hand, if σ is too large, the positional information of the oligomers is completely ignored and every oligomer that occurs in both sequences contributes equally to the similarity score of the sequences, regardless of its positions.

2.1.2 Radial Basis Function Kernel

In addition to the string kernels, there is a widely used kernel function to measure the similarity between two samples, namely the radial basis function (RBF) kernel. Thanks to its ability to capture the underlying non-linear and complex patterns in the data, it is used to measure the similarities in the data from different domains. Let x and y be two numeric vectors of the same length. One can consider these vectors as feature vectors with the same set and order of attributes representing samples in a dataset. The RBF kernel measures the similarity between these samples as follows:

$$K(x,y) = exp\left(-\frac{\left\|x-y\right\|^2}{2\sigma^2}\right) \tag{2.2}$$

where σ is a parameter that regulates the level of similarity. As a common representation, one can reformulate Equation 2.2 by setting $\gamma = \frac{1}{2\sigma^2}$ as follows:

$$K(x,y) = exp\left(-\gamma\left\|x-y\right\|^2\right) \tag{2.3}$$

To calculate the expression $\left\|x-y\right\|^2$ in both equations, one has to perform elementwise subtraction of the feature vectors x and y, and calculate the norm of the resulting vector. Even though such a computation is easy to perform in plaintext, computing this operation while preserving privacy is difficult and inefficient, if not infeasible. Therefore, this work utilizes the privacy-friendly reformulation of the RBF kernel function using the dot product operation:

$$K(x,y) = exp\left(-\gamma(\langle x,x\rangle - 2\langle x,y\rangle + \langle y,y\rangle)\right) \tag{2.4}$$

where $\langle,\rangle$ represents the dot product operation. Thanks to Equation 2.4, one can compute the similarity between x and y using only the dot products $\langle x,x\rangle$, $\langle x,y\rangle$ and $\langle y,y\rangle$.

2.1.3 Support Vector Machines

One of the most successful and widely used ML methods, especially for small to medium size data, is the SVM [16]. It is a supervised machine learning method that seeks linear separation of the samples by a hyperplane that maximizes the margin between the borderline samples of each class, called *support vectors*. The following optimization problem yields the desired hyperplane:

$$\min_{w}\left(\frac{1}{n}\sum_{i=1}^{n}\max\left(0,1-y_i(w^T x_i - b)\right) + \lambda\left\|w\right\|^2\right) \tag{2.5}$$

where w is the normal vector of the hyperplane, n is the number of samples, x_i is the i-th sample whose class label is y_i, b is the offset and λ is the parameter adjusting the trade-off between maximizing the margin and the correctness of the side of x_i.

The given form of the SVM optimization function in Equation 2.5 allows only the linear classification of the samples in their original space, which is also called *input space*. In addition to the hard margin version, there is a soft margin version which adjusts the trade-off between the correct labeling and margin size. However, linear separation of the samples in the input space is not always possible or the performance is not sufficient. Kernel functions are at utmost utility for a good separation. They enable mapping the samples from the input space to another, usually higher dimensional space, which is called *feature space*, so that the separation of the samples from different classes can be achieved by a hyperplane in this space. To integrate the kernel functions into the optimization problem in Equation 2.5, one needs to replace each dot product with the kernel function which is equivalent to the dot product in the feature space: $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$ with mapping function $\Phi : X \rightarrow \mathbb{R}^m$. This gives SVMs the ability to capture more complex patterns by separating of the samples which is not necessarily linear in the input space but linear in the feature space.

In order to train the SVM algorithm, one can use the pre-computed kernel matrix K, which is a positive semi-definite square matrix. The SVM training algorithms can extract the required dot product of the samples from this kernel matrix and find a hyperplane providing the desired separation among the samples from different classes. This thesis exploits the idea of using a pre-computed kernel matrix to train the SVM algorithm in order to preserve the privacy of the samples used to train the algorithm by computing the desired kernel matrix privately.

2.1.4 Recurrent Kernel Networks

DNNs have become very popular in parallel with the increase in computational power. There are now several different types of DNNs to process differently structured data. Images, for example, are mostly processed by convolutional neural networks (CNN). They are able to capture the patterns in images by employing convolutional layers, maxpooling layers and fully connected layers to name just a few of the options. For sequential data, such as genomic sequences, recurrent neural networks (RNNs) come in handy. These networks can identify the recurring patterns in the sequential data using *memory*, which can be short- or long-term. RNNs work effectively, especially when there is enough data to learn the patterns. However, it is difficult to find an effective architecture of RNN that works well on genomic data as opposed to the well-known string kernels, which have been shown to perform well specifically for this type of data. In order to bridge the gap between the RNN and string kernels, Chen et al. [17] proposed a special way to construct an RNN whose internal computation is the same as that of the substring kernel allowing mismatches and the local alignment kernel, which are used in several studies proving their utility [18, 19, 20].

As an overview, Chen et al. [17] designed an RNN whose internal computation is the recursive computation of the above kernel functions, which can be computed using dynamic programming [21]. Unlike the traditional kernel functions that require prior selection of hyperparameters, the proposed RNN-based computation of kernel functions, which they call recurrent kernel network (RKN), benefits from the backpropagation algorithm to optimize these parameters. This feature of RKN saves trying a high number of different combinations of these parameters by hand in the traditional grid search approach. Based on the results in their paper, RKNs outperform the traditional substring kernel and local alignment kernel, as well as LSTMs [22]. This suggests that RKNs have high potential on various problems from different domains.

In a broader perspective on RKNs, they employ k-mer-like small motifs, called *anchor points*, to measure how similar the sequences are. They act like different templates that one tries to fit onto the sequences to measure the similarities among different sequences based on how much these templates are fitting onto them. The anchor points have a similar encoding to one-hot encoding with a slight difference that a character of an anchor point is not necessarily a single character in the alphabet. Instead, the encoding of a character of an anchor point bears the probability of it being each character in the alphabet.

Since this thesis is concerned with the prediction part of RKNs, the details about the backpropagation are omitted and only the forward pass needed to make predictions is given. Let q be the number of anchor points with k characters, let d be the length of the vector encoding these characters, let x be a sequence with s characters, each of which is encoded via one-hot encoding vectors of length d. The similarity between a character of the input sequence, x_t, and a character of an anchor point, z_j^i, is computed as follows:

$$K(x_t, z_j^i) = e^{\alpha(\langle x_t, z_j^i \rangle - 1)} \tag{2.6}$$

where α is a free parameter. Such a similarity computation is performed for the j-th character of all anchor points, yielding a vector of length q, $b_j[t]$. The process continues with the following computation:

$$c_j[t] = \lambda c_j[t-1] + c_{j-1}[t-1] \odot b_j[t] \tag{2.7}$$

where $c_j[t]$ represents the initial mapping of the sequence up to the t-th character into a q-dimensional vector based on anchor points of length j and λ is a scalar value decreasing the effect of the previous time points for $j \in \{1, \ldots, k\}$ and $t \in \{1, \ldots, s\}$. Stopping conditions for this recursive computation are $c_0[t]$ and $c_j[0]$ which are a vector of 1s and a vector of 0s if $j \neq 0$, respectively.

Once the initial mapping of the sequence for $j \in \{1, \ldots, k\}$ is obtained, the RKN proceeds with the multiplication of $c_k[j]$ by the orthogonalization factor $K_{Z_j Z_j}^{-1/2}$, which is the inverse square root matrix of the Gram matrix of the anchor points up to their j-th character. As Chen et al. [17] also noted, this is the only non-standard component of the entire computation of RKN. In substring kernels allowing mismatches, the final step of the forward pass to obtain the prediction of the input sequence is to apply the linear layer, which basically performs a dot product between $c_k[s]$ and the weight vector of the classifier w. This completes the forward pass for a single input sequence and the prediction of a test sequence on the pre-trained RKN model.

2.2 Area Under the Curve

One of the most widely used performance evaluation metrics for ML algorithms is the area under curve (AUC). It summarizes the output of plot-based evaluation methods as a single value. Briefly, it calculates the area under the curve of the desired method's plot. This thesis investigates AUC computation on the receiver operating characteristic (ROC) curve and the precision recall (PR) curve. In general, if the area under the curve is 1, it indicates that the model performs well. Specifically for the AUC of ROC, the area around 0.5 is an indication of a model predicting randomly.

2.2.1 Area Under Receiver Operating Characteristic Curve

One of those two plot-based evaluation methods is the ROC curve. It is used to evaluate how well a ML model with binary outcome performs by considering the sensitivity and the specificity of the model, which are calculated using the prediction confidence value (PCV) of the test samples. The ROC curve plots the false positive rate (FPR) on the x-axis and the true positive rate (TPR) on the y-axis, which are calculated using PCVs. A well-performing model would have a plot in which TPR becomes close to 1 before FPR increases. In order to summarize an ROC curve, AUC measures the area under this curve using the following formula:

$$AUROC = \sum_{i=1}^{N} \Big(T[i] \cdot (F[i] - F[i-1]) \Big) \tag{2.8}$$

where N is the number of test samples that are sorted in descending order in terms of PCVs, $T \in [0,1]^N$ and $F \in [0,1]^N$ are the TPR and FPR values of the corresponding samples, respectively. For the correct result, Equation 2.8 assumes that there is no tie between the PCVs of test samples. This means that every PCV is unique. If this is not the case, then Equation 2.8 approximates the results based on the order of the test samples. Depending on the case, this could result in very different results compared to the correct result. For an illustrating example, the reader can refer to Paper VI.

To address the correct calculation of the AUC in case of a tie in PCVs, one can use the following formula:

$$AUROC = \sum_{i=1}^{|I|} \Big(T[I[i-1]] \cdot (F[I[i]] - F[I[i-1]]) + \frac{(T[I[i]] - T[I[i-1]]) \cdot (F[I[i]] - F[I[i-1]])}{2} \Big) \tag{2.9}$$

where I denotes the index vector of test samples in ascending order and $|I|$ represents the size of the vector. I contains the indices of PCVs whose value is different from the preceding PCV. By this formula, it is possible to compute the exact AUC in case of a tie in PCVs.

2.2.2 Precision Recall Curve

The second plot-based evaluation method addressed in this thesis is the PR curve. Similar to the ROC curve, the PR curve also evaluates the models with binary outcomes. It plots the recall on the x-axis and the precision on the y-axis. Based on this plot, one can assess the performance of a model. For instance, a well-performing model should have a curve that starts with a precision of 1 and goes without too much drop until the recall becomes 1. In order to ease this evaluation, AUC can be used so that the result of the evaluation becomes a single scalar value. An AUC close to 1 indicates a well-performing model.

The computation of the AUC of PR curve requires a similar computation as the AUC of ROC curves with tie, because the precision and recall values change even if there is no tie in the PCVs. Therefore, assuming that T is the precision and F is the recall, Equation 2.9 is employed to calculate the AUC of the PR curve.

2.3 Cryptographic Techniques

This section introduces the cryptographic techniques used in this thesis to make the ML algorithms private.

2.3.1 Randomized Encoding

One of the cryptographic techniques used in this thesis is randomized encoding (RE) [23]. It is a masking technique that uses random masks to protect the privacy of the data during computation.

Compared to homomorphic encryption (HE), another crytographic technique in the literature, it is much lighter and faster without compromising privacy. The idea of RE is to keep the masked input values secret except for the desired output. It masks the input values by purposefully adding/subtracting/multiplying random values to/from/with those input values. In this way, the party receiving these masked input values can extract only the output of the function that the owner of the secret value wants to reveal from these masked input values by combining them in a certain way.

Applebaum [24] proposed the RE of several basic operations. The first and most basic of these is the addition function $f(x_1, x_2) = x_1 + x_2$. The encoding of this function is as follows:

$$\hat{f}(x_1, x_2; r) = (x_1 + r, x_2 - r) \tag{2.10}$$

where r is a uniformly chosen random value. Decoding of this encoding to obtain the desired output is simply done by adding $x_1 + r$ and $x_2 - r$. This addition cancels the random value used to mask x_1 and x_2, and reveals only the addition of x_1 and x_2, but nothing else about these secret values to a third-party that only receives the results of $x_1 + r$ and $x_2 - r$.

They also gave the encoding of the multiplication function $f(x_1, x_2) = x_1 \cdot x_2$. In order to encode this function, they proposed the following:

$$\begin{aligned} \hat{f}(x_1, x_2; r_1, r_2, r_3) &= (x_1 + r_1, x_2 + r_2, r_2 x_1 + r_3, r_1 x_2 + r_1 r_2 - r_3) \\ &= (c_1, c_2, c_3, c_4) \end{aligned} \tag{2.11}$$

where r_1, r_2 and r_3 are uniformly chosen random values and c_1, c_2, c_3 and c_4 are the components of the encoding. In order to decode this encoding and obtain the result of the function $f(x_1, x_2)$, one needs to compute $c_1 \cdot c_2 - c_3 - c_4$. This cancels the random values used and gives the desired output $x_1 \cdot x_2$ without getting to know anything about x_1 or x_2.

In addition to the addition and multiplication operations, there is also a function combining these operations, namely the multiplication-addition function $f(x_1, x_2, x_3) = x_1 \cdot x_2 + x_3$. The encoding of such a function is achieved as follows:

$$\begin{aligned} \hat{f}(s_1, s_2, s_3; r_1, r_2, r_3, r_4) &= (s_1 - r_1, r_2 s_1 - r_1 r_2 + r_3, s_2 - r_2, r_1 s_2 + r_4, s_3 - r_3 - r_4) \\ &= (c_1, c_2, c_3, c_4, c_5) \end{aligned} \tag{2.12}$$

where r_1, r_2, r_3 and r_4 are uniformly chosen random values and c_1, c_2, c_3, c_4 and c_5 are the components of the encoding. In order to recover the output of f from $\hat{f}$, one needs to compute $c_1 \cdot c_3 + c_2 + c_4 + c_5$. This computation cancels out the random values and gives $x_1 \cdot x_2 + x_3$ without revealing anything about the input values.

In addition to the above functions, Applebaum [24] also stated that any arithmetic circuit with logarithmic depth can be encoded using RE. This thesis benefits from this ability of RE to encode the dot product of two vectors of any length later on. More details about the algorithm generating an encoding for an arithmetic circuit can be found in [24] and the details of the algorithm generating encoding for the dot product function can be found in Paper III in the Appendix.

2.3.2 Multi-party Computation

Another cryptographic technique employed in this thesis to make ML algorithms private is multi-party computation (MPC) [25, 26]. In short, in MPC, multiple parties participate in the computation of a function with private input value(s) such that these parties should not learn about the input value of any of the other parties and their share of the result of the function. In other words, the parties should only know their input values and, in the end, their own share of the output of the function.

In order to achieve privacy of the secret values in MPC, one of the key components is the secret sharing of the data. None of the parties in MPC should have all the shares. These shares must be shared among the parties involved in the computation in such a way that they do not reveal any information about the actual value. There are several schemes in the literature for sharing the secret values among the computing parties. This thesis employs two different secret sharing schemes and introduces only these schemes to focus on the required preliminaries for the papers. One of them is the arithmetic secret sharing scheme. In arithmetic secret sharing, there are different methods to arithmetically share a secret value, but the common feature of all these methods is that the secret value is split into two or more seemingly random shares, so that a single share has no meaning by itself, but the addition of these shares over a ring gives the secret value. Among these methods, this thesis uses 2-out-of-2 additive secret sharing in which a secret value x is shared between two parties over a ring $\mathcal{Z}_R$ such that the addition of the shares x_1 and x_2 yields x over this ring. Even though the specific notation may vary in the papers using MPC with 2-out-of-2 additive secret sharing, $\langle x \rangle_i^R$ represents the share of x over $\mathbb{Z}_R$ at the i-th party.

The second secret sharing employed by this thesis is boolean secret sharing. In this scheme, a secret value x is split into two secret shares x_1 and x_2 such that $x_1 \oplus x_2$ yields x where $\oplus$ is the XOR operation. In general, throughout this thesis the boolean sharing of x in the i-th party is shown as $\langle x \rangle_i^B$ but, similarly, this may vary in individual papers. Readers are encouraged to refer to the notations given in each paper for a more accurate representation of the shares of a secret value in that paper.

2.4 Privacy Preserving Machine Learning Approaches

In the literature of privacy preserving ML, there are many attempts to realize different ML algorithms in a privacy preserving way using different cryptographic techniques. One of these techniques is differential privacy (DP) [27]. The core idea of DP is to introduce noise into the computation based on a budget so that the exact values of the data used in the computation can be preserved. There are several ways to perturb the data and output in the computation. When noise is added to the training data, this approach is called input perturbation. A common application of this approach is to perturb the input data for a specific function so that the result is within the expected range of errors caused by the use of noisy data. The limiting factor in this approach is that perturbing the input data for all functions is very difficult, if not impossible, due to the variety of functions that can be computed with the data. In addition to the input perturbation, there are objective function perturbation [28] and output perturbation. They aim to provide privacy by introducing noise into the objective function and the output of the function, respectively. Due to the nature of DP, all these approaches cannot provide the exact result of the computation. However, in some fields and applications, such as precision medicine, it might be important to obtain an exact result. Since the goal of this thesis is to propose the exact private computation of the functions, the studies employing DP are out of the scope of this thesis.

2.4.1 Privacy Preserving SVM

Since an SVM requires pairwise similarity of samples in terms of dot products in the same space for training, it cannot be made private by iterative approaches easily. Therefore, it is necessary to gather the whole samples in one place to train an SVM algorithm. To achieve this, there are several approaches in the literature that use different cryptographic techniques. Vaidya et al. [29] proposes an approach to privately train an SVM algorithm on horizontally shared data by computing the kernel matrix without compromising the privacy of the data. Their approach assumes that the samples are represented as binary feature vectors such that each attribute of the samples can be either 0 or 1. They benefit from the idea that the dot product of two binary vectors can be computed by counting the intersecting 1s of these vectors. They achieve this private counting using secure set intersection cardinality [30], which gives the number of 1s on the same indices in both vectors. Considering the wide range of applications that need to work with real numbers, this feature of their proposed approach limits its use quite considerably. Therefore, it is not widely applicable.

Zhang et al. [31] proposed an approach that addresses this missing part to some extent. In their approach, they aim to privately compute the dot product between the feature vectors of samples, which are not necessarily binary, from different sources on a third-party and train an SVM algorithm at the end. To achieve the privacy requirements, they use integer vector encryption. Each data source encrypts its vectors with its own key and sends the encrypted vectors to the third-party. By using these encrypted vectors coming from different data sources, the third-party can compute the encrypted dot product of them. To obtain the result of this dot product operation in plaintext, the key of the dot product of the vectors, each encrypted by their owners with different keys, has to be switched to a key that the outsourced third-party knows. To do this, they employ the key-switching technique [32]. This key only reveals the dot product of these vectors, but nothing else about them. Although this approach works with integer vectors, it is not extendable to vectors of real numbers. Moreover, it is not efficient and scalable due to its encryption scheme. It cannot handle the computation when the number of samples increases. More details about this issue can be found in Appendix I.

In addition to the key-switching based solution, Liu et al. [33] also proposed an encryption based approach that enables mining of the outsourced data using the SVM algorithm. They consider a scenario where the data is outsourced after encryption, and the owners of the data and the cloud, to which the data is outsourced, jointly mine the encrypted data without compromising privacy. They use fully homomorphic encryption (FHE) to meet privacy requirements and allow several operations, such as addition and multiplication, on the encrypted data. However, due to the usage of FHE, the proposed approach is not efficient enough to train an SVM. Moreover, they perform communication between the cloud and the data owners, which further reduces the efficiency of the approach.

2.4.2 Privacy Preserving Deep Learning

In addition to the studies that provide the privacy preserving computation of the SVM algorithm, there are several studies in the literature that propose approaches for privacy preserving computation for various deep learning algorithms. These studies are generally based on MPC or HE, and mostly target DNNs and CNNs.

SecureML, proposed by Mohassel and Zhang [34], is one such study that aims to make deep learning algorithms private. It uses MPC to ensure the privacy of the samples during the training and testing of the algorithm. In their setup, they employ 2 parties and classify their framework as 2-party computational framework. Using these 2 parties, they compute several basic functions that serve as building blocks for more complex operations. These building blocks can work not only with integers, but also with positive and negative decimal values thanks to the number format used. This number format is a fixed-point arithmetic that represents values with a fixed number of bits. Certain number of least significant bits are for representing the fractional part, the most significant bit is for the sign, and the rest are for the integer part. In their study to demonstrate the applicability of their proposed approach, they employed these building blocks to realize the private training and inference on logistic regression as well as a fully connected neural network. They train a 3-layer DNN in which the exact rectified linear unit (ReLU) value and the privacy friendly version of the softmax activation functions are computed. After training, they also perform privacy preserving inference on the trained model using their proposed building blocks.

Inspired by SecureML, Wagh et al. [2] develop a more efficient framework based on MPC, namely SecureNN, which offers basic building blocks for more complex operations such as ReLU, maxpool, normalization and so on. Since these operations in deep learning algorithms need to be performed with decimal values, they inherit the fixed-point arithmetic from SecureML so that they are able to represent positive and negative decimal values in their computations. One of the distinguishing features of SecureNN from SecureML is that they use 3 computing parties, making their framework a 3-party computational framework as opposed to a 2-party computation of SecureML. This additional party in the computation allows some of the computations to be performed more effectively. Moreover, Wagh et al. [2] targeted not only fully connected neural networks, but also more complex DNN architectures such as CNNs. They implemented the training and inference of several different CNN architectures without compromising the privacy of the data.

In addition to MPC-based approaches, there are attempts in the literature using HE to perform secure computation of deep learning algorithms. One such study is proposed by Bakshi and Last [35], which targets RNN. In their study, they introduced an approach for secure inference on an RNN model trained on plaintext data with somewhat HE. They tackled two major issues that arise with HE, which are the accumulation of noise through the consecutive operations and the computation of non-linear activation functions. Due to the nature of HE, the consecutive secure operations accumulate errors on the encrypted data, which may end up with a completely wrong result. To deal with this problem, they presented three different ciphertext refresh mechanisms. Besides the problem of error accumulation, HE can only allow addition and multiplication on the encrypted data. They are inadequate for the non-linear activation functions used in deep learning algorithms, which are the majority of them. To address this issue, they come up with two different solutions. One of them is to outsource the computation of the non-linear activation functions to the owner of the data. This requires the owner to be active the whole time during the computation and brings additional communications between the owner and the computing party. The second solution is to approximate the non-linear activation functions with polynomial functions that can be computed on encrypted data. Such an approximation sacrifices to some extent the performance of the model to be able to compute the activation functions inherently. Even with these solutions that sacrifice the performance of the model, the execution time is significantly higher than the plaintext inference.

Paper	Crypto. Technique			Algorithm			Application				
	RE	SE	MPC	SVM	RKN	AUC	HIV	GE	SCOP	AML	UCI
RE-based framework	✓			✓			✓				
RE-based framework on gaze estimation	✓			✓				✓			
ESCAPED		✓		✓			✓				
Privacy preserving SVM with MPC			✓	✓			✓				
CECILIA			✓		✓				✓		
ppAURORA			✓			✓				✓	✓

Table 2.1: Summary of the studies presented in this thesis. This table summarizes the topics of the papers in terms of crytographic techniques used, algorithms addressed and problem applied. The cryptographic techniques are randomized encoding (RE), special encoding for dot product (SE) and multi-party computation (MPC). The algorithms are support vector machines (SVM), recurrent kernel networks (RKN) and area under the curve (AUC). The applications are HIV coreceptor usage prediction (HIV), gaze estimation (GE), structural classification of proteins (SCOP), acute myeloid leukemia (AML) and UCI heart disease benchmark dataset (UCI).

2.5 Necessity of Efficient Privacy Preserving Machine Learning Algorithms

There is still room for improvement in the privacy preserving ML. In the privacy preserving SVM algorithm, it is necessary to find efficient and applicable solutions for training and testing. The solution should work with all numeric types. It should also be able to perform training and testing in a privacy preserving way without sacrificing the performance of the model in terms of execution time and accuracy. As for privacy preserving deep learning, there are also points that can be improved. For example, the exact computation of several activation functions is still lacking. To get the full performance of a deep learning algorithm in privacy domain, these activation functions should be privately computable. Furthermore, the complexity of the deep learning algorithms has been increasing and training such networks is challenging even in the plaintext domain. This leads to the need for more efficient, scalable, secure and private deep learning frameworks that are capable of realizing these networks while protecting privacy. This thesis aims to bridge this gap by proposing different mechanisms for the private computation of various ML algorithms without sacrificing the accuracy of the computation or the privacy of the data used in the computation. Table 2.1 summarizes the studies included in this thesis in terms of the cryptographic techniques used to provide the privacy and security, the machine learning algorithms desired to be made private and the applications of the resulting privacy preserving machine learning algorithm.

3 Objectives and Expected Outcomes

The goal of this thesis is to address the issues raised at the end of Chapter 2 and contribute to the privacy preserving ML literature by introducing a complete efficient and accurate privacy preserving ML pipeline. To achieve this goal, privacy preserving training of an ML algorithm, one of the key components of the pipeline, occupies an important place in this thesis. We aim to train various ML algorithms, especially those that have not been well-studied or effectively solved, without sacrificing the performance of the model or the efficiency of the solution. The privacy preserving training of kernel-based ML algorithms are among the targeted ML algorithms since they are relatively difficult to deal with in the privacy domain due to the requirement to compute the kernel function between all pairs of samples. In addition to the training phase of the pipeline, we also address the testing phase for several different ML algorithms. In addition to the privacy preserving kernel-based ML algorithms, we aim to perform inference on a model without sacrificing the privacy of the data or the model. Moreover, we also target privacy preserving inference for modern neural network architectures such as RKNs. Just as important as the training and inference phase of the ML pipeline, there is another substantial component of the pipeline, namely the privacy preserving evaluation of a model, that we aim to address in this thesis. We are concerned with the collaborative evaluation of the performance of a model without sacrificing the privacy of the data, not even the labels. We expect to perform the evaluation without revealing the sensitive information of the samples to anyone except the owner. In order to achieve these goals, we aim to develop special cryptographic techniques that allow the computation of ML algorithms without breaking the privacy, and to introduce new approaches based on existing techniques. We seek not only theoretical, but also practical and efficient solutions to the dilemma between the need to protect the privacy of data and the desire of ML algorithms for data in the training, testing and evaluation phases of the ML pipeline.

4 Results and Discussion

As we have seen in Chapter 2, there is still room for improvement in the ppML domain. Some of the ML algorithms have not been addressed before and some are the subject of this thesis. For example, RKNs have not yet been realized in a privacy preserving manner and are one of the subjects of this thesis. Compared to other traditional deep learning algorithms, RKNs are relatively complex to compute due to their non-standard orthogonalization factor. In addition to the unaddressed ML algorithms, the solutions dealing with privacy preserving computation of some ML methods, such as SVMs and other kernel-based ML methods, lack efficiency and practicality. Furthermore, the collaborative evaluation of the performance of models is not well-studied. The literature of privacy preserving ML algorithms still demands for efficient and practical solutions for these algorithms and their evaluations. All these issues harm the idea of a complete privacy preserving ML pipeline. To advance the field and fill this gap, this thesis aims to address the ML algorithms that have not been concerned previously with known cryptographic techniques and to propose novel and efficient ways of computing the ML algorithms that have not been solved effectively. This thesis also aims to realize the collaborative evaluation of model performance without compromising the privacy. In this section, I summarize the results of the works of my Ph.D. research towards this direction and discuss these results in terms of their soundness as well as their contribution to the literature of privacy preserving ML algorithms. To provide a better overview, I group the studies according to the targeted ML algorithm and the cryptographic technique used. We begin with our first attempt to enable the privacy preserving computation of the kernel-based ML algorithms using an RE-based framework and the applications of this framework in the fields of precision medicine and gaze estimation. We then continue with the privacy preserving RBF kernel computation using a specially designed encoding for the dot product and MPC-based privacy preserving SVM training and inference, and their applications to precision medicine problems. In the last section, we present a general purpose MPC framework, called CECILIA, for more complex operations and the applications of this framework in privacy preserving AUC computation and privacy preserving inference on pre-trained RKNs.

4.1 Privacy Preserving Computation of Kernel-based Machine Learning Algorithms with an RE-based Framework and Their Applications

One of the distinguishing features of the kernel-based ML algorithms is the pairwise similarity of the samples to train and test these algorithms. This makes it difficult to handle them in a privacy

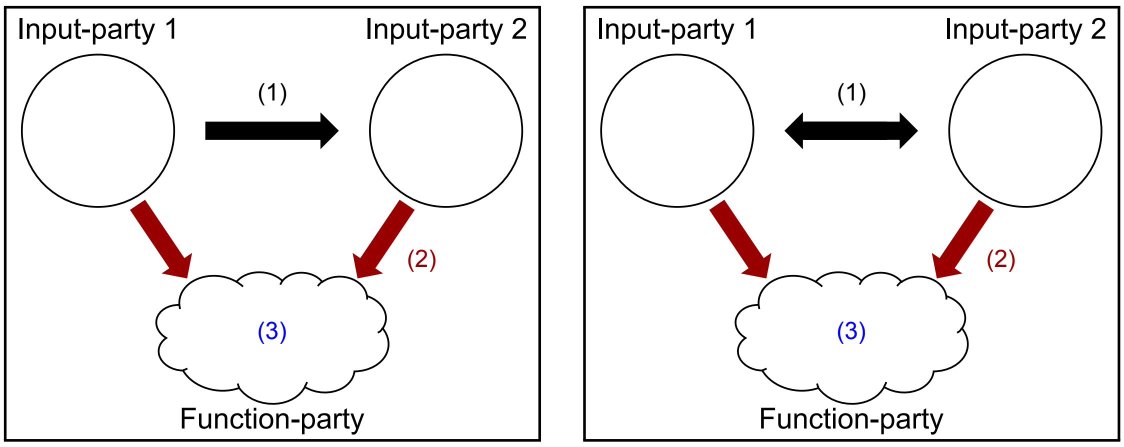


Figure 4.1: The overview of the computation in RE-based framework is summarized in two different figures. **(a)** The computation of the dot product of vectors from two input-parties is shown. (1) At first, the input-party 1 shares the random values with the input-party 2. (2) Then, both input-parties compute the required components of the encoding for the element-wise multiplication of vectors in the function-party and send these components to the function-party. (3) At the end, the function-party obtains the element-wise multiplication and eventually the dot product of these vectors. **(b)** In the oligo kernel computation, everything except the communication between the input-parties is the same. The communication among these input-parties is not unidirectional as it is the case in the dot product computation. It is rather bidirectional, that is the input-party 2 is also sending required masked data to the input-party 1.

preserving way. In this section, we see our first attempt at the privacy preserving computation of the kernel-based ML algorithms by utilizing a masking technique, called RE. We introduce a framework and take a look at the applications of this framework to precision medicine and gaze estimation problems. This section is based on Paper I and Paper II given in the Appendix.

4.1.1 Privacy Preserving SVM for Precision Medicine

As the first attempt towards one of the goals of this thesis, which is to realize ML algorithms in a privacy preserving way, we conducted a study to privately train and test kernel-based ML algorithms eventually leading to privacy preserving SVM training and inference. This subsection is based on Paper I in the Appendix.

SVMs have proven to produce excellent results in a variety of domains and problems. They are capable of separating samples from different classes with a hyperplane. Since such separation may not be ideal for every space from which samples originate, SVMs are often accompanied by kernel functions. SVMs can employ kernel functions corresponding to implicitly mapping samples from their input space, where the samples are not well separable with a hyperplane, to a different and generally higher dimensional space, where the samples are better separable with a hyperplane. This makes SVMs very versatile and powerful encouraging privacy preserving training and inference on it to emerge. One of the scenarios that requires privacy preserving training and testing SVMs is when two sources decide to outsource the collaborative training and testing of SVMs to an external computing party. In this scenario, the privacy of the samples has to be protected. No other than the designated parties are allowed to see the samples and/or the result of the computation in plaintext. To enable this privacy preserving computation in the above scenario, we developed a framework for computing the kernel matrix based on several kernel functions by using RE, a masking technique, to ensure the privacy of the samples during the computation. The computed kernel matrix can then be used to SVM training and inference or in general the kernel-based ML algorithms by the third-party.

In this study, we aim to compute the privacy preserving SVM with an RBF kernel and an oligo kernel. These kernels require the computation of the dot product between the samples and the positional differences of the occurrence of k-mers in sequences from different sources on an external computing party, respectively, without compromising the privacy of the samples. We refer to these sources as *input-party* and the external computing party as *function-party*.

To compute the kernel matrix based on the oligo kernel, we need to have the relative differences of the occurrences of all k-mers over an alphabet between each pair of sequences in the process. To achieve this, we take advantage of RE addition given in Equation 2.10. This encoding allows the input-parties to encode the occurrences of the k-mers in their sequences in such a way that the function-party can only learn the differences between these occurrences, but nothing else.

To compute the kernel matrix based on the RBF kernel, we use the privacy-friendly version of the RBF kernel given in Equation 2.4, which requires the dot product of the vectors. To obtain the dot product, we benefit from RE multiplication of two values. We first perform the element-wise multiplication of two vectors from different input-parties on the function-party without revealing the actual values of the vectors. Then, the function-party can calculate the addition of the elements of the element-wise multiplication of these vectors to obtain the dot product of these vectors. Afterwards, the function-party employs the resulting dot product matrix, or Gram matrix in the input space, to compute the RBF kernel matrix. It is important to note that the Gram matrix can be used to compute other kernel matrices that can be computed by the dot products of the vectors.

Once the desired kernel matrix is obtained, the function-party can train the SVM algorithm and infer on the resulting model without revealing the samples of an input-party to the other input-party or the function-party.

In order to demonstrate the correctness and applicability of our proposed approach, we selected the HIV coreceptor prediction problem. It is an important precision medicine problem since, in the treatment of HIV patients, it is crucial to determine which coreceptor HIV uses to enter human cells, so that these coreceptors of cells can be blocked by known drugs and HIV can be prevented from duplicating in the cell [36]. In our study, we conducted experiments to solve this prediction problem without breaking the privacy. We simulated two input-parties by splitting the data into two sets. Then, we performed the computation of the kernel matrices with both the RBF kernel and the oligo kernel by employing the encoded samples from these sets on the function-party. Afterwards, we trained and tested an SVM model using the resulting kernel matrices separately. Without sacrificing the privacy of the samples, we computed the exact kernel matrices that can be obtained with no privacy constraints by using the same set of samples, which then leads to the exact same performance of the SVMs. We evaluated the prediction performance of both the private and the non-private models using the area under receiver operating characteristic curve (AUROC) and both gave the same result for both kernel functions. Since the oligo kernel computation has not been addressed so far, we only compared the RBF kernel computation with the key-switching based privacy preserving SVM [31] approach. Since both approaches compute the same kernel matrix and therefore result in the same AUROC, we focused on comparing the execution time of the approaches. This comparison showed that our RE-based approach is significantly more efficient than the key-switching based approach.

4.1.2 Privacy Preserving Gaze Estimation

In addition to the HIV coreceptor usage prediction problem from the field of precision medicine, we also applied the RE-based framework for privacy preserving kernel-based ML algorithms to the gaze estimation problem. Besides the applicability of the framework to a different domain, this study demonstrates that the enable other kernel-based ML algorithms in a privacy preserving way. This subsection grounds on Paper II.

The eye movements of individuals contain useful information about them. The developments in virtual reality (VR) have improved both the quality and quantity of eye movement data. This data can be used for various tasks such as diagnosing diseases [37] and determining human intention [38]. Therefore, it is important to protect the privacy of this data during processing. In this study, we aimed to demonstrate the privacy preserving prediction of gaze after the extraction of the landmark data of the eye images. To meet the privacy requirements during the computation of the kernel matrix indicating the similarity between samples, we used the RE-based framework for privacy preserving kernel-based ML algorithms. Finally, we trained and tested the support vector regression (SVR) algorithm, which is a variation of the SVM for continuous labels, to predict eye gaze based on the resulting kernel matrix.

In this study, we generated 20.000 synthetic eye images using UnityEyes [39] and extracted 36 eye landmark features to train and test the SVR algorithm. Then, we split the data into two input-parties to simulate the input-parties in the RE-based framework. Once these input-parties had their data, they encoded their samples so that the function-party could only obtain the dot product of these samples, from which it could compute the RBF kernel matrix, but nothing else. Once the function-party computed the kernel matrix, it proceeded to train the SVR algorithm to estimate the gaze of the eye images based on the extracted features without compromising the privacy of the data used in this process.

The result of this study showed that it is feasible to perform the gaze estimation task while maintaining privacy. We succeeded in predicting gaze from eye images without sacrificing the performance of the model or the privacy of the data. We evaluated the model using the mean angular error for different number of samples and found that it is 0.21, 0.18 and 0.17 for 5.000, 10.000 and 20.000 samples, respectively. These results are exactly the same as those that could be obtained without privacy concerns. This shows that the proposed framework is able to achieve the same performance while preserving privacy.

This study was the first to bring together gaze estimation and privacy protection based on RE. By enabling privacy preserving gaze estimation with our RE-based framework, we demonstrated that the framework is able to handle different problems from various domains. It is not only limited to the field of precision medicine, but also applicable to other domains where the kernel-based ML algorithms can be used. Moreover, the utilization of kernel-based SVR is also an important contribution to the privacy preserving ML literature. The proposed framework works not only with SVMs or other kernel-based classification algorithms, but also with kernel-based regression algorithms such as SVR. Besides the theoretical contribution, this work also demonstrated that private gaze estimation can be practically solvable. The framework is capable of training the SVR algorithm privately and performing privacy preserving inference on the resulting model quite efficiently. Such efficiency is also substantial because it shows that our system can achieve real-time privacy preserving inference of gaze on VR devices. We managed to perform privacy preserving inference of a single eye image in about 1.125 milliseconds.

4.1.3 Overview

The correctness and applicability of the RE-based framework was shown using two different problems, namely HIV coreceptor usage prediction and gaze estimation. The experiments on these problems showed that the framework enables to compute the same kernel matrix and provide the same performance that can be computed and obtained without privacy. Moreover, the experiments demonstrated that the framework is not limited to kernel-based classification algorithms. Instead, it can be employed to perform kernel-based regression, such as SVR, while maintaining privacy. Additionally, the framework enables performing the prediction of the label of a sample in a reasonably short period of time. The execution time of the proposed framework showed that it is efficient enough to be used in real-life cases. Besides all these positive features of the RE-based framework, it lacks the ability to include additional input-parties in the computation. The framework is limited to only two input-parties. If more than two input-parties are involved in the computation, all the samples of all input-parties will be revealed to the function-party because of the way the dot product of these samples is computed. In the proposed framework, the dot product is computed by first calculating the element-wise multiplication of the samples privately and then summing these multiplications. With more than two input-parties, the function-party would have $v_1 \otimes v_2$, $v_1 \otimes v_3$ and $v_2 \otimes v_3$ where $\otimes$ is the element-wise multiplication operation and v_i is a feature vector of a sample belonging to the i-th input-party. Based on the results of these element-wise multiplications, the function-party could easily determine the actual values of all feature vectors. Considering these issues, there was still room for improvement in the literature of privacy preserving kernel-based ML algorithms. There was a need to find effective solutions that allow more than two input-parties in the private computation of these algorithms, which is what we addressed in the following research.

4.2 More Efficient RBF Kernel Computation for Privacy Preserving SVM

In order to address the problem that the previous framework was unable to include more than two input-parties due to privacy concerns, we introduced two studies that allow more than two input-parties to participate in the computation without compromising data privacy. These studies use different cryptographic techniques to ensure data privacy. Paper III and Paper IV in the Appendix form the basis for this subsection.

4.2.1 ESCAPED: Efficient Secure and Private Dot Product Framework

The lack of ability of the previous framework, which we presented in Section 4.1, to include more than two input-parties in the computation encourages us to find a new solution to the private dot product computation that is flexible about the number of input-parties in the computation. Efforts

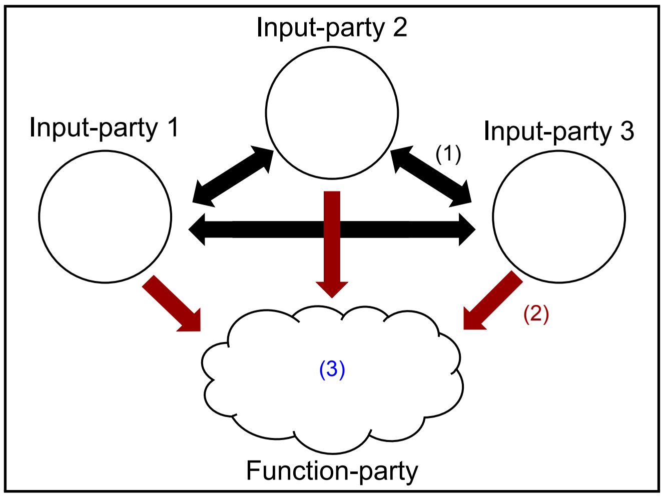


Figure 4.2: Overview of the computation in ESCAPED is depicted. **(1)** At first, the input-parties share the required random values and masked data between each other in order to create the components of the encoding. **(2)** After computing these components, the input-parties send these components to the function-party. **(3)** Once the function-party receives these components, it combines them in a certain way to obtain the dot product between the data of every pair of the input-parties.

toward this goal lead to a new framework, called ESCAPED. We base this subsection on Paper III in the Appendix.

In most cases, computing the dot product between vectors from different input-parties is not necessarily limited to a scenario with two input-parties. In practice, there could be more than two input-parties willing to participate in the joint training and testing of a kernel-based ML algorithm. To enable this process, dot products of the feature vectors of all input-parties are required. Since the previous framework cannot incorporate more than two input-parties due to the privacy leakage that would occur when there are more than two input-parties in the computation, we designed a new and special encoding scheme for the dot product operation that allows more than two input-parties. Based on this encoding scheme, we created a new framework, namely ESCAPED. In ESCAPED, we privately compute the Gram matrix of the data from two or more input-parties on the function-party, and then train and test a kernel-based ML algorithm while maintaining privacy.

We devised a new lightweight encoding scheme that allows the private computation of the dot product of two matrices of real numbers, whose columns are individual samples from different input-parties on the function-party. In this scheme, each pair of input-parties has a separate communication to compute the dot product of their matrices. In each pair of input-parties, the input-parties mask their input matrices with matrices of uniformly selected random values. The individual components of this encoding do not reveal anything unless they are combined in a designated way. Along with the dot product of the matrices with themselves, the input-parties send these components to the function-party. The result of the combination of these components yields the dot product of two matrices from different input-parties in the function-party. Once every pair of input-parties performed this computation, the function-party obtains all dot products of matrices allowing the construction of the global Gram matrix in a privacy preserving manner. Afterwards, the function-party computes the desired kernel matrix, and trains and tests a kernel-based ML algorithm. As a major improvement over the previous framework, ESCAPED is capable of including more than two input-parties in the computation without compromising the privacy of the data of these input-parties thanks to its special encoding scheme for the dot product. This efficient encoding scheme allows the input-parties to perform the private dot product computation in pairwise manner. This means that in the case of N input-parties involved in the computation, there must be $\binom{N}{2}$ different communication pairs as well as sets of random matrices.

We demonstrated the correctness and effectiveness of ESCAPED using the same problem to which we applied the previous framework, namely the HIV coreceptor usage prediction problem in the field of precision medicine. To evaluate the results of the experiments, we used AUROC and F1-score. They show that with ESCAPED we obtained the same AUROC and F1-score that could be obtained with the same set of parameters and the samples when there is no privacy concern. This demonstrated that ESCAPED is capable of computing the exact kernel matrix as the non-private computation could result in. The same kernel matrices then led to the same SVM model, whose performance was the same for both private and non-private settings. Since the performance of both ESCAPED and the previous framework was the same, we compared them in terms of their execution time. For this comparison, we conducted an experiment with a scenario where we only had two input-parties, since the RE-based framework works only with two input-parties. The result of this experiment revealed that ESCAPED is significantly faster than the previous framework thanks to its novel and specific encoding scheme for the dot product computation. Considering that RE can be used to encode any logarithmic depth arithmetic circuit [24], we also compared ESCAPED with the RE-based approach where we specifically compute the dot product of vectors from different sources as opposed to the previous framework, where we computed the element-wise multiplication of vectors in the function-party and the function-party sums these multiplication results to obtain the dot product of these vectors. Due to the large number of random values in the RE-based approach resulting in a large number of components in the encoding, ESCAPED outperformed the RE-based approach in terms of execution time. In addition to the comparisons, we analyzed the scalability of ESCAPED to a varying number of input-parties in the computation. We conducted experiments with $2, 3, 4, 5$ and 6 input-parties, respectively. The results demonstrated that the execution time of ESCAPED is empirically between linear and quadratic, but close to being linear.

In addition to the supervised learning kernel-based ML algorithms, to demonstrate the applicability of ESCAPED to unsupervised learning, we also used ESCAPED to stratify cancer patients into clinically meaningful subgroups via multi-omics dimensionality reduction and clustering without sacrificing patient privacy. For this purpose, we replicated the study conducted by Röder et al. [40]. Using the same set of parameters they used, we computed the kernel matrices of the patients via ESCAPED based on different multi-omics, resulting in different views of the patients. To perform the clustering of the patients into subgroups after the computation of the kernel matrices, we used web-rMKL [40] as done by Röder et al. [40]. We evaluated the results by survival analysis and showed that it leads to the same p-value. This demonstrates that ESCAPED is not limited to supervised kernel-based ML algorithms. It can also be applied to unsupervised kernel-based ML algorithms, as well as to any computation that requires the dot product of matrices from different sources.

These results of ESCAPED show that the computation of the dot product of vectors from two or more input-parties on an external third-party is feasible without compromising the privacy of the vectors unlike the previous RE-based framework. Even though the RE-based approach can compute the dot product of vectors from more than two input-parties, it is not efficient. In terms of execution

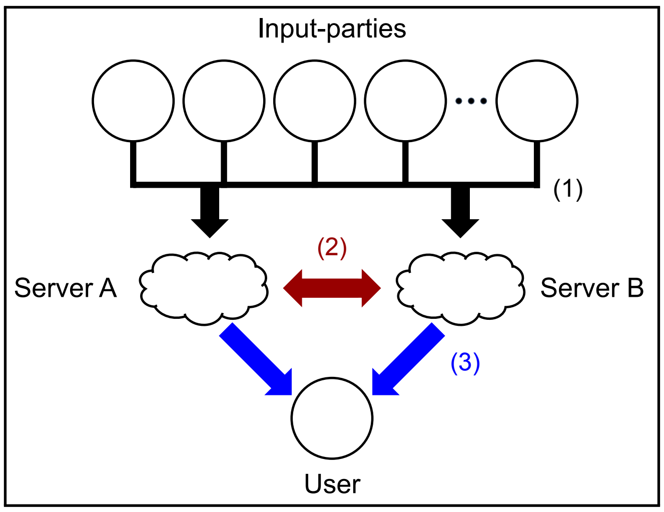


Figure 4.3: Overview of the computation in MPC-based SVM. **(1)** The input-parties outsource their data to the servers in a secret shared form. **(2)** Then, the servers compute the required functions in order to compute the desired kernel matrix. **(3)** At the end, the servers send the share of the desired kernel matrix that they have to the designated user.

time, ESCAPED performs this computation quite efficiently and outperforms the RE-based approach, which is based on the state-of-the-art lightweight encoding in the literature, thanks to the utilized novel encoding scheme that is specific to the dot product computation. This encoding scheme is a significant contribution to the privacy preserving machine learning literature. Moreover, this study is the first to show that the privacy preserving multi-omics dimensionality reduction and clustering, which has recently been shown as the state-of-the-art approach in the literature [41], is feasible. This also demonstrates that ESCAPED can also be used to perform kernel-based clustering. Besides all these achievements and novelties, there are also some drawbacks of ESCAPED. Although it is possible and relatively efficient to include additional input-parties in the computation, it is still challenging to perform the computation when many input-parties are involved in the computation. Since there must be separate communication between each pair of input-parties to determine and exchange the random matrices used to encode the input matrices, ESCAPED could not efficiently handle the case where there are too many input-parties. Furthermore, the communication between the input-parties to encode their input matrices requires that these parties are actively involved in the computation. For instance, if a new input-party participates in the computation, that new input-party must communicate with all other input-parties already involved in the computation so that the function-party can compute the corresponding parts of the kernel matrix. This requirement for input-parties hurts the practicality of ESCAPED to some extent. Moreover, such pairwise communication between input-parties may not be possible at all for various reasons such as privacy, technical difficulties, etc. These drawbacks of ESCAPED bring us to the next paper, which uses MPC to address these concerns while keeping the advantages of ESCAPED to some extent.

4.2.2 Privacy Preserving SVM via MPC

To more effectively handle the addition of new input-parties into the privacy preserving training and testing of the SVM algorithm, we propose a solution based on MPC. This subsection is based on

Paper IV in the Appendix.

Since too many input-parties in ESCAPED can become a challenge due to the required pairwise communication between the input-parties to exchange the random values for the encoding, we devised a new MPC-based solution to the privacy preserving computation of the dot product of vectors from different input-parties. Thanks to the nature of MPC, it is much easier to include additional input-parties in the computation. Furthermore, unlike ESCAPED, the input-parties do not need to be active during the computation or when adding a new input-party to the computation. This makes our MPC-based solution more applicable to real-life scenarios. In addition to the advantages over ESCAPED, our MPC-based solution also retains to some extent the advantages of ESCAPED. The execution time of our solution remains low and acceptable in real-life applications.

In this study, we used the 2-PC setting, in which data sources outsource their data to two computing parties, namely servers, which perform the computation of the dot product of the outsourced vectors from multiple input-parties. Each of these servers has a share of the vectors of the input-parties. In order to ensure the privacy of the data of the input-parties, we used two different secret sharing techniques, namely the arithmetic secret sharing and the boolean secret sharing. In the version with arithmetic secret sharing, the servers can compute the dot product of secret shared vectors of positive integers by first computing the privacy preserving element-wise multiplication of these vectors and then adding the resulting shares of the multiplications to obtain the share of the dot product of the vectors. In this version of the solution, each positive integer is represented by a set of bits depending on the ring size and this requires working with more than a single bit for each value. However, when dealing with binary data, the boolean secret sharing is more practical. In boolean secret sharing, each value is either 0 or 1 and the representation of these values can be done by a single bit. To address this particular case, we proposed another version of the privacy preserving dot product computation for binary vectors based on boolean secret sharing. We first performed the privacy preserving *AND* operation and then the secure addition of the resulting vector to count the number of 1s on the same indices in both vectors to obtain the dot product of the secret shared vectors. Once the Gram matrix of vectors is computed, it is given to the user to enable the training and testing of the SVM algorithm on the user side while maintaining the privacy of the input data.

To demonstrate the correctness of the proposed solution, we applied it to the HIV coreceptor usage prediction problem. Since the sequences are encoded by one-hot encoding, which results in binary vectors, we can use both secret sharing techniques. We compared the results of these experiments with the non-private versions of these experiments using the same set of parameters. For both secret sharing techniques, the comparisons show that our MPC-based solution is able to compute the exact same Gram matrix and performs the same without compromising privacy. In addition to the correctness analysis, we also performed the scalability analysis of our proposed solution. To do this, we conducted experiments on synthetic data using both secret sharing techniques. The results showed that our solution scales between linear and quadratic to the number of samples, the number of features and the number of data sources, but is close to linearity empirically. We compared our MPC-based solution with the key-switching based privacy preserving SVM [31] and this comparison shows that our approach is significantly more efficient with both secret sharing techniques. When we compared the execution time of the secret sharing techniques, the boolean secret sharing outperformed the arithmetic secret sharing, suggesting that one can proceed with the boolean secret sharing when the data is binary.

The results of the experiments demonstrated that MPC, in particular 2-PC, can be used to privately compute the Gram matrix of samples belonging to different input-parties, which then allows training and testing of the SVM algorithm on the user side. The user receives the resulting Gram matrix in plaintext and computes the desired kernel matrix to train and test the SVM algorithm without sacrificing model performance or data privacy. While the MPC-based solution is still efficient in terms of execution time, it is more effective than ESCAPED when there are many input-parties or a new input-party is involved in the computation. Besides these advantages, this solution has a significant drawback. It cannot work with negative values or decimal values. It is limited to integers when the arithmetic secret sharing is used, and only to binary values when the boolean secret sharing is used. Considering that most of the ML algorithms work with both positive and negative real numbers, the utility of the proposed approach is limited due to its restricted capabilities in the type of numbers it can work with.

4.2.3 Overview

In this section, we see that the efficient privacy preserving computation of the Gram matrix, more precisely the dot product of samples from different input-parties on a third-party or two computing parties, is feasible. The resulting Gram matrix can then be used to compute the desired kernel matrix to train and test the SVM algorithm or another kernel-based ML algorithm on the same third-party or an external user. Compared to the framework we present in Section 4.1, we managed to improve the efficiency of the computation of the dot product. Furthermore, we are now capable of including more than two input-parties in the training and testing of the kernel-based ML method. As an improvement on the previous RE-based framework, ESCAPED and the MPC-based approach can work with more than two input-parties without compromising the privacy of samples. In ESCAPED, the new encoding scheme designed specifically for the dot product operation enables ESCAPED to compute the Gram matrix of the samples of multiple input-parties on the function-party in a privacy preserving way. This encoding scheme allows efficient computation when there are a small number of input-parties, but it is difficult for ESCAPED to handle computation with a relatively large number of input-parties. Moreover, the input-parties are supposed to be active during the computation. The MPC-based solution, on the other hand, can effectively cope with the large number of input-parties and newly added input-parties without requiring the input-parties to be active. Since the execution time of the MPC-based solution is comparable to that of ESCAPED, this practicality of MPC encourages us to continue working with it to address the unresolved issues, such as working with both positive and negative real numbers and performing other arithmetic operations that are widely used in ML algorithms.

4.3 Towards a General Purpose MPC Framework for More Complex Operations

Even though the specific encoding for the private computation of a function works slightly faster than the MPC-based solution for the kernel-based ML algorithms, the requirement of the input-parties to be active during the computation, the communication among the input-parties, the inefficiency of adding a new input-party to the computation and the ineffective handling of a large number of input-parties in the computation encouraged us to proceed with MPC as a privacy preservation technique to address other operations used in ML algorithms. In this section, we introduce a generic

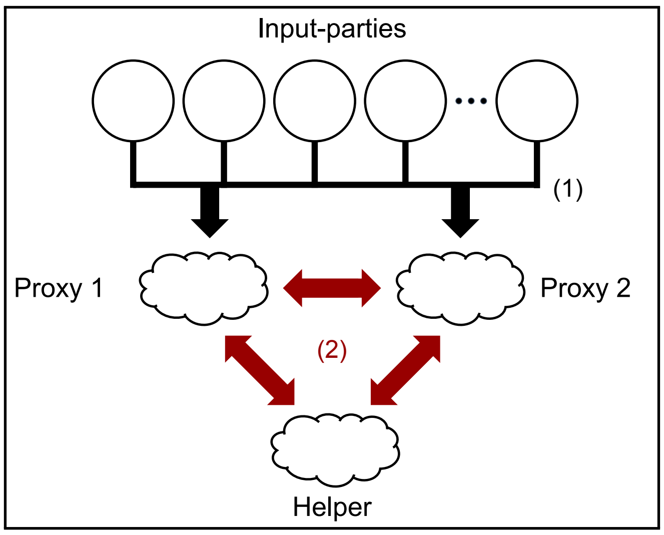


Figure 4.4: Overview of the computation in CECILIA. **(1)** The input-parties outsource their data to the proxies in a secret shared form. **(2)** Then, the proxies compute the required functions with the help of the helper party.

framework based on MPC that provides several secure basic operations to perform more complex functions in a privacy preserving way. The details are described in Paper V and Paper VI in the Appendix.

4.3.1 Comprehensive Secure Machine Learning Framework

Considering the flexibility to add new data sources to the computation and the effectiveness in handling a large number of data sources, we aim to design an MPC-based generic framework with several basic building blocks such as addition, multiplication, multiplexer, exponential, etc., so that we can address more complex and broad ML algorithms in a privacy preserving way. This subsection grounds on Paper V in the Appendix.

So far, we have targeted specific ML learning algorithms, such as kernel-based machine learning algorithms and SVMs, using somewhat specific encoding schemes which are difficult, if not impossible, to adapt to some of the other ML algorithms. This makes the previous works limited to a specific set of ML algorithms. However, there are several effective and powerful ML algorithms that the privacy preserving ML literature lack. To bridge this gap, we designed an MPC-based generic framework, called CECILIA, that provides several secure building blocks which are privately performing several operations used in ML algorithms. We then utilized these secure building block to realize these ML algorithms without sacrificing the privacy of data or the privacy of the model. Using CECILIA, it is now possible to compute some of the ML algorithms in a privacy preserving way, some of which have not been addressed previously. We used CECILIA to perform privacy preserving inference on a specific RNN, called RKN [17].

We based CECILIA on 3-party computation. Two of these computing parties are called *proxy*, which are the parties to which the data sources outsource their data in a secret shared form, and the third party is called *helper*, which, as the name implies, helps the proxies perform the desired

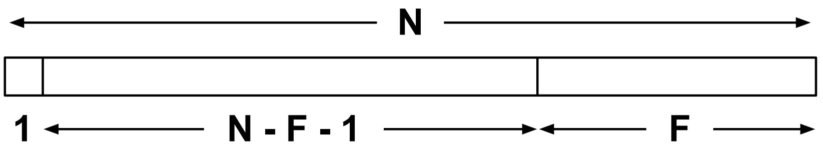


Figure 4.5: Number format used in ESCAPED. Let N be the number of bits used to represent a value. The most significant bit indicates the sign of the value. The least F bits are allocated to the fractional part of the value. The rest $N - F - 1$ bits are used to represent the integer part of the value. As an illustrating example, let N be 6 and F be 2. In this setting, 6.25 is represented as 011001 and −1.75 is represented as 111001 in this number format.

operation in a privacy preserving manner.

Since many ML algorithms work not only with integers, but also with positive and negative decimal values, we adapt the number format used by [34] in CECILIA. This number format, which is depicted in Figure 4.5, uses a fixed number of bits to allow representation of both negative and positive real values up to a certain precision in a binary format. Similar to two's complement, the most significant bit is allocated for the sign of the value we want to represent. If it is 1, it means that the value is negative. If it is 0, the value is positive. Depending on the parameter F of CECILIA, F-many least significant bits are assigned to represent the decimal part of the value. The rest of the bits are for representing the integer part of the value. This number format allows CECILIA to work with positive and negative real numbers up to a certain precision, enabling CECILIA to fulfill the condition of many ML algorithms about the number type.

In addition to the number type condition, the framework should provide sufficient coverage of basic operations so that it can be used to implement ML algorithms in a privacy preserving way. CECILIA offers the privacy preserving versions of several basic operations, some of which are also included in other frameworks. These common operations are the addition of two secret shared values and the multiplication of two secret shared values. We also provide efficient versions of some previously addressed operations, such as the oblivious selection among two secret shared values and the identification of the most significant bit of a secret shared value. In addition to these common basic building blocks, CECILIA also introduces some novel privacy preserving operations. One of these novel operations is the exponential of a secret shared value. CECILIA is able to compute the exponential of a base, which is known by both proxies in plaintext, raised to the power of a secret shared value. We are inspired by the square-and-multiply algorithm for computing the exponential of a base raised to a positive integer power. We extend the idea of the square-and-multiply algorithm so as to cover not only the positive integer values represented in binary form, but also the positive and negative real values represented in the number format of CECILIA. The second novel operation we introduce in CECILIA is the privacy preserving computation of the inverse square root of a secret shared Gram matrix (INVSQRT). One of the methods for computing the inverse square root of a Gram matrix is to first perform the eigenvalue decomposition of the matrix and then compute the reciprocal of the square root of the eigenvalues, and construct the desired matrix using these eigenvalues and the original eigenvectors. We follow the same approach, but in a privacy preserving manner. To accomplish this, we adapt the approach of outsourcing the eigenvalue decomposition [42] to our 3-PC scenario so that we have the masked eigenvalues and the eigenvectors in the helper party. After the helper party has the masked eigenvalues and eigenvectors, it performs the sharing of the masked eigenvectors without any constraint. The proxies can unmask the share of the masked eigenvectors and obtain the share of the original eigenvectors. However, with respect to the eigenvalues, the helper party takes a special approach to sharing. It purposefully divides the share of the eigenvalues between the proxies so that they can perform the square root and reciprocal operations on the shares of the eigenvalues. Once they obtain the share of the eigenvectors and the reciprocal of the square root of the eigenvalues, they reconstruct the inverse square root of the Gram matrix using the secure multiplication and addition operations of CECILIA and obtain the share of the desired matrix.

We employed these building blocks to realize the privacy preserving inference on a pre-trained RKN model. This was the first time that RKN inference was performed in a privacy preserving manner. We refer to this application of CECILIA as ppRKN. We split the model parameters between the proxies in arithmetic secret sharing form and give them the share of a test sample on which the proxies can perform the required operations in a privacy preserving way to obtain the prediction of this sample. During this process, neither of the computing parties learns anything about the input data, nor does the data owner obtain any information regarding the model parameters.

We analyzed ppRKN in terms of its correctness and scalability to various factors affecting execution time. To demonstrate the correctness of CECILIA, we used the same problem as Chen et al. [17], which is the Structural Classification of Proteins (please refer to Paper V for more details). It consists of several fold recognition tasks. We trained the RKN algorithm on a randomly selected task and performed inference on the model in plaintext using randomly chosen samples from the test set. To compare the result of the plaintext inference with the privacy preserving inference, we first shared the model parameters between the proxies. Afterwards, the proxies performed the required operations for inference using the building blocks of CECILIA and reconstructed the resulting prediction score for comparison. The comparison showed that we managed to compute almost exactly the same prediction score except for some precision error, which is less than 2×10^{-5}. The error stems from the fact that we only have a limited number of bits to represent values in our system, which causes some precision loss that can accumulate. Note that we performed the empirical correctness analysis on LAN since WAN has no effect on the correctness of the results.

As opposed to the correctness analysis, the network type has a significant impact on the execution time of ppRKN. Therefore, we analyzed the impact of the number of anchor points, the length of k-mers and the length of the input sequence on the execution time of ppRKN on both LAN and WAN. In both settings, the analyses demonstrated that ppRKN scales linearly with the length of k-mers and the length of the input sequence. It also scaled almost linearly with the number of anchor points. When we compared the results between the network types, WAN took more time to compute , as expected, due to the relatively high round trip times between the computing parties.

The correctness analysis demonstrated that CECILIA is capable of realizing the privacy preserving inference on a pre-trained RKN model without revealing the sensitive information of the input sequence to the computing parties and the model parameters to any party involved in the computation. Considering that the complexity of the operations used in RKNs is higher than the operations in most of the deep learning algorithms, CECILIA has a high potential to enable not only the RKN inference, but also other deep learning algorithms in a privacy preserving manner. Furthermore, the applicability of CECILIA in real-life scenarios seems feasible, as demonstrated by the analysis of the scalability of CECILIA to various parameters of the RKN method.

One of the most important outcomes of this study was that a generic framework to realize the

ML algorithms in the privacy domain is feasible. Instead of having a unique and separate privacy scheme for each ML algorithm, it is practical to have a generic framework that provides the ability to realize any ML algorithm or at least most of the ML algorithms privately. The success of ppRKN proves that CECILIA is an important step towards the goal of having such a framework. CECILIA offers some novel privacy preserving operations that have not been addressed in the literature before. Thus, CECILIA covers the set of ML algorithms better than other privacy preserving ML frameworks in the literature such as SecureML [34] SecureNN[2].

Up to this point in the thesis, we managed to privately train several ML algorithms and perform inference on a model without violating data privacy. However, another important part of the complete privacy preserving ML pipeline is missing, which is the collaborative evaluation of the model in a privacy preserving manner.

4.3.2 ppAURORA

To obtain a complete privacy preserving ML pipeline, we use the building blocks of CECILIA to address one of the least studied parts of this pipeline, namely the privacy preserving evaluation of a ML model. We base this subsection on Paper VI in the Appendix.

In the previous works, we have addressed the privacy preserving training of several ML algorithms and the privacy preserving inference on several pre-trained algorithms. However, the privacy preserving collaborative evaluation of a ML model of the complete privacy preserving ML pipeline is lacking. If multiple data sources want to perform a collaborative testing of an ML model and the privacy of the labels of the test samples of these sources matters, the data sources cannot outsource the collaborative evaluation of the model by simply sharing the labels and the predictions of the test samples with a third-party in plaintext. The collaborative evaluation has to be performed privately to ensure the protection of sensitive information, which is, in this case, the labels of the samples and the prediction scores. In order to address this need, we use CECILIA to propose the privacy preserving computation of one of the widely used evaluation metrics in ML studies, namely the AUC. It summarizes the information of a curve-based evaluation of an ML model as a single value. There are several plot-based evaluation metrics in the literature. In this study, we address the private collaborative computation of the area under the ROC curve with and without tie conditions in the predictions (AUROC) and the PR curve (AUPR).

The computation of AUC requires the predictions of all samples that are sorted based on their prediction confidence values (PCVs). Assuming that the PCVs of the samples from each individual data source are sorted, the first task of the computing parties is to obtain a globally sorted PCVs of all test samples from all data sources in a privacy preserving way. This process can be thought of as the merging step of the merge sort algorithm, where individually sorted lists are merged to obtain the globally sorted list. To achieve this in a privacy preserving way, we merged the lists in pairs. We compared the top PCVs of two lists and selected the larger one while protecting the privacy of the samples. In this sorting, the computing parties cannot associate the elements between the globally sorted list and the individually sorted lists. Even though this privacy preserving merging of the individually sorted lists provides full privacy, it requires relatively many rounds of communication between the computing parties. In order to have a more practical privacy merging option, we make this process parametric so that it becomes significantly faster by only exposing the mapping of δ-many samples from the individually sorted list to the globally sorted list, where δ is the parameter that adjusts the trade-off between practicality and privacy. Once the globally sorted list is available, the computing parties proceed with the AUC computation of the desired evaluation metric.

In privacy preserving AUROC computation, the computing parties proceed to calculate the number of true positive (TP) and false positive (FP) samples for each split of the sorted list. In the without-tie version of AUROC, the computing parties treat the PCV of each sample as a different threshold and privately calculate the nominators and the denominators of the TPR and FPR, separately, which are required to compute the AUC of the ROC curve. Afterwards, the computing parties iterate over these values of each sample to compute the nominator and denominator of AUROC. The final step is to divide these values in a privacy preserving manner to obtain the AUROC without tie. When using the with-tie version of AUROC, the computing parties first privately determine the samples after which PCV changes and treat them as threshold PCVs. Then, the computing parties follow the same procedure as the AUROC without-tie version, but in the AUROC with-tie version, only the TP and FP values of these threshold PCVs have an impact on the result and the rest are neutralized during the process without knowing which PCVs are thresholds or have been neutralized.

The area under the PR curve is also a often-used performance metric for ML methods, especially when labels are unbalanced. Since the PR curve can be computed similarly to the ROC curve with tie condition, we employ the computational approach of AUROC with-tie for the privacy preserving computation of the area under the PR curve (AUPR). The only difference is that the denominators of the precision for each sample are different and this forces the use of the privacy preserving division operation to compute the precision of each sample in the list, as opposed to a single private division operation at the end in AUROC.

We demonstrated the correctness of the AUC computation on two real datasets up to a certain precision, which results from the adjustable parametric precision of the privacy preserving division operation. The first dataset is the Acute Myeloid Leukemia dataset of the first subchallenge of a DREAM Challenge [43] and the second is the UCI Heart Disease benchmark dataset [*]. In both datasets, we obtained the exact AUC for both ROC and PR curves up to a certain precision as one could obtain without privacy. This shows that we can compute the same exact AUC in a privacy preserving way. In addition to the correctness analysis, we also analyzed how ppAURORA scales to different parameters in the computation. More specifically, we conducted experiments to understand the impact of the number of data sources, the number of samples in the data sources and the δ parameter on the execution time of ppAURORA. The analyses demonstrated that ppAURORA scales between linear and quadratic to the number of data sources with a fixed number of samples in each data source and the number of samples in a fixed number of data sources. ppAURORA manages to perform both privacy preserving AUROC and AUPR computations of $1,000$ samples from each 16 data source in about $2,500$ seconds. With respect to the δ parameter, which adjusts the trade-off between the privacy and practicality of ppAURORA, we find that the execution time displays a logarithmic decrease when δ increases. As an example of the choice of δ that balances the trade-off between privacy and practicality, ppAURORA takes only 150 seconds to compute the AUC of 1000 samples from each 8 data source when we set δ to 11.

The results of the experiments demonstrated that we are able to privately and efficiently compute the exact same AUC of ROC and PR curves up to a certain precision as one could obtain without privacy constraint. This ability addresses the private and collaborative evaluation part of the com-

[*] https://archive.ics.uci.edu/ml/datasets/heart+disease

plete privacy preserving ML pipeline. It is now possible to collaboratively evaluate how well a model performs without compromising the privacy of the data used in any of these phases.

4.3.3 Overview

This section shows that an MPC-based generic framework to realize the ML algorithms in a privacy preserving way is feasible without inserting noise into the computation or approximating the operations. With CECILIA, we are able to perform several basic operations with a small precision error due to the number format that we use to represent numbers in our system. In addition to the basic operations such as addition and multiplication, to the best of our knowledge, we addressed the exact exponential computation and the exact computation of the inverse square root of Gram matrix for the first time in the literature. In addition to these operations, in CECILIA we improve the computational efficiency of the already solved operations, such as the multiplexer and the most significant bit. We used these building blocks of CECILIA to realize privacy preserving inference on a specific type of RNNs, namely RKNs. Compared to other DNNs, RKNs are difficult to train and test. They have a non-standard operation, which is the inverse square root of a Gram matrix, on top of the standard operations such as addition, multiplication, exponential, etc. The implementation of the private inference RKN indicates that CECILIA can be used to address other DNN types in a privacy preserving way. Furthermore, CECILIA provides more operations with higher computational accuracy compared to the other privacy preserving ML frameworks that use approximation for the activation functions. In addition to the privacy preserving inference, we employed the building blocks of CECILIA to address the collaborative evaluation of a model in a privacy preserving way. We developed an application of CECILIA, called ppAURORA, to calculate the AUC of two different widely used curve-based evaluation metrics, which are ROC curve and PR curve. With only a small precision error, we privately obtained the exact same AUC of both metrics. The computation of AUC shows that we are capable of performing not only the training and inference of the model, but also the collaborative evaluation of the model without sacrificing the privacy of the data. In summary, CECILIA and its application contribute to the privacy preserving ML literature and represent a major step towards a complete privacy preserving ML pipeline.

5 Conclusion

This thesis promotes the idea of a complete privacy preserving ML pipeline. It addresses the training, testing and evaluation of an ML algorithm without sacrificing the privacy of the data or model parameters.

To achieve this, the thesis introduces several studies that address different phases of the pipeline. The first study, presented in Section 4.1.1, uses a lightweight encoding scheme, namely RE, to ensure the privacy of data from two different input-parties that want to train and test a kernel-based ML method, such as an SVM and SVR, on a function-party without exposing their data to another party in the computation. In that study, the input-parties allowed the function-party to obtain the Gram matrix of their data in order to train and test a kernel-based ML algorithm without knowing anything about the data of the input-parties. This framework was applied to several problems from different domains such as precision medicine (refer to Section 4.1.1) and gaze estimation (refer to 4.1.2) in order to demonstrate the ability to solve them in a privacy preserving manner.

Even though the proposed RE-based framework provides privacy protection for a specific scenario, it lacks the flexibility to extend to more than two input-parties. The studies introduced in Section 4.2.1 and Section 4.2.2 addressed this issue in order to improve the ability to compute kernel-based ML algorithms such as SVMs in a privacy preserving way so that more than two input-parties can participate in the computation. The first of these studies presented a special lightweight encoding for private computation of the dot product of multiple sources on the third-party to obtain a Gram matrix and eventually train and test a kernel-based ML algorithm privately. This new encoding scheme cannot efficiently handle adding new input-parties and too many input-parties altogether, which was addressed in the second study. It employs MPC to ensure privacy and sacrifices execution time slightly for the sake of effectively handling a large number of input-parties and new input-parties participating in the computation.

Despite the effective solution for collaborative computation of Gram matrices in a privacy preserving way, it is not feasible to address every single ML algorithm to accomplish the goal of a complete privacy preserving ML pipeline. Instead, a generic privacy preserving ML framework can be more effective, which is what Section 4.3.1 introduces. Considering the effectiveness of MPC in dealing with input-parties, that section proposes an MPC-based comprehensive secure machine learning framework, CECILIA, which provides several privacy preserving basic operations to realize many ML algorithms in a privacy preserving manner. CECILIA was used to address the privacy preserving inference of a complex deep learning method, more specifically RKNs. It is the first study in the literature to perform the privacy preserving inference of RKNs.

In addition to the efforts in privacy preserving training and testing of ML algorithms, the study in Section 4.3.2 used the building blocks of CECILIA to address the privacy preserving collaborative evaluation of an ML model. It occupies an important place in the complete privacy preserving ML pipeline and the application of CECILIA to the evaluation phase, ppAURORA, contributes to the privacy preserving ML literature.

The studies introduced in this thesis make important contributions to the idea of a complete privacy preserving ML pipeline in many different ways. They show how an existing cryptographic technique can be used to implement a previously unaddressed ML algorithm in a privacy preserving manner. They also present a new cryptographic technique, specifically designed for private computation of the dot product to address kernel-based ML algorithms. Moreover, they also prove that an existing cryptographic technique can be employed to design the exact privacy preserving computation of functions, which either had not been addressed before or were only calculated by approximation. As an important contribution towards the complete privacy preserving ML pipeline, this thesis addresses the privacy preserving collaborative evaluation of an ML model via AUC as well.

This thesis contains scientifically important and successful studies that address various aspects of a complete privacy preserving ML pipeline using several different novel and existing cryptographic techniques. However, there is still room for improvement. One of the most important missing phases of a complete privacy preserving ML pipeline is privacy preserving data preprocessing. It occupies an important place in ML studies. Furthermore, the existing function coverage of CECILIA is not sufficient to address every ML algorithm in the literature while protecting privacy. CECILIA needs to be extended to allow the privacy preserving training and testing of other ML algorithms that we have not addressed in this thesis. Although the efficiency of CECILIA is promising, the privacy preserving ML literature could require even more efficient solutions given the increasing complexity of ML methods like deep learning methods and the increase in dataset sizes.

6 Appendix

In this chapter, all the publications used in this thesis are given with minor template modifications. They have the permission to be included in this dissertation. The definitive versions of the publications can be found in the respective venues. The corresponding citations of the publications are listed in Chapter 1 in the same order in which the publications appear here.

I A framework with randomized encoding for a fast privacy preserving calculation of non-linear kernels for machine learning applications in precision medicine

Ali Burak Ünal Mete Akgün Nico Pfeifer

Abstract

For many diseases it is necessary to gather large cohorts of patients with the disease in order to have enough power to discover the important factors. In this setting, it is very important to preserve the privacy of each patient and ideally remove the necessity to gather all data in one place. Examples include genomic research of cancer, infectious diseases or Alzheimer's. This problem leads us to develop privacy preserving machine learning algorithms. So far in the literature there are studies addressing the calculation of a specific function privately with lack of generality or utilizing computationally expensive encryption to preserve the privacy, which slows down the computation significantly. In this study, we propose a framework utilizing randomized encoding in which four basic arithmetic operations (addition, subtraction, multiplication and division) can be performed, in order to allow the calculation of machine learning algorithms involving one type of these operations privately. Among the suitable machine learning algorithms, we apply the oligo kernel and the radial basis function kernel to the coreceptor usage prediction problem of HIV by employing the framework to calculate the kernel functions. The results show that we do not sacrifice the performance of the algorithms for privacy in terms of F1-score and AUROC. Furthermore, the execution time of the framework in the experiments of the oligo kernel is comparable with the non-private version of the computation. Our framework in the experiments of radial basis function kernel is also way faster than the existing approaches utilizing integer vector homomorphic encryption and consequently homomorphic encryption based solutions, which indicates that our approach has a potential for application to many other diseases and data types.

I.1 Introduction

By the recent development of next generation sequencing (NGS) technologies, DNA sequencing and RNA sequencing can be performed effectively and efficiently [44]. However; the generated sequence data contains private information about the host and one could infer many phenotypes of an individual, such as hair color, skin color and more importantly genetic diseases, from the relevant sequence data of that individual [45, 46, 47]. Such private information can be used against the owner of the sequence data potentially leading to increased health insurance premiums due to the genetic disease of that person. On the other hand, machine learning algorithms still require these sequence data to capture the underlying patterns of the diseases. In many of the real-world problems, machine learning algorithms need to have more data than one source can provide [48, 49, 50].

In this paper, we consider a scenario where we have two parties with sequence data, each of which we call *input-party* and one party, which we call *function-party* that wants to run machine learning algorithms on the data of these input-parties. To avoid the aforementioned privacy issues in our scenario due to the leakage of sequence data utilized in machine learning algorithms, we propose a framework utilizing randomized encoding [23, 24] to enable machine learning algorithms involving a single type of basic arithmetic operation to use the data from two sources without sacrificing the privacy of participants. We provide the function-party with the private calculation of four basic arithmetic operations, that is addition/subtraction and division/multiplication. In order to demonstrate the performance of our framework, we chose support vector machine (SVM) with the oligo kernel and the radial basis function kernel on the prediction of coreceptor usage of HIV based on V3 loop sequences [51]. We computed the kernel matrices by employing our proposed framework and trained a prediction model on top of these kernel matrices.

In the rest of the paper, we will explain the similar studies in the literature in Section I.2. We will give the background information about the oligo kernel, the radial basis kernel and randomized encoding in Section I.3. Then, we will propose our framework in Section I.4. Afterwards, we will discuss the security of the proposed framework in Section I.5. Next, we explain the dataset that we utilized in the experiments in Section I.6. We will then show and evaluate the results of these experiments in Section I.7. Finally, we will conclude the paper in Section I.8.

I.2 Related Work

In this context, we refer to a source with data, such as a clinic having sequence data, or an entity computing a function, such as a university conducting a study on the data of clinics, as party. In the literature, some approaches using an SVM are based on the distributed model, which assumes that each party of the computation has its own data and the function needs to be calculated by using the data of all parties. Vaidya et al. [29] proposed a privacy-preserving SVM classification algorithm. In the proposed approach, each party has its own data and they compute the gram matrix to train an SVM model by using a modified secure dot product calculation method. However; it focuses more on binary feature vectors and does not support advanced string kernels. Furthermore, there are also a number of studies which employ an outsourced model in which the data of the parties are stored on a cloud server as encrypted by the secret key of the owners of the data. Liu et al. [52] introduced an SVM algorithm which can be used to mine the encrypted outsourced data. Since the data is encrypted before outsourcing, this slows down the process of model training. Zhang et al. [31] also proposed a secure dot product calculation method to train an SVM model. The underlying idea is to transform the secret key of the dot product of two vectors encrypted by different keys into a known key by the server. In order to accomplish the transformation, the server collaborates with the owners of the keys of these vectors. The proposed approach fits into our scenario where we have two input-parties and one function-party. However; the approach utilizes integer vector encryption to preserve the privacy of the data. Therefore, it cannot handle the data having a large number of features within a reasonable time frame.

I.3 Preliminaries

In this section, we will explain the kernel functions that we utilized and the randomized encoding which is our base security scheme.

I.3.1 Oligo Kernel

Although the oligo kernel belongs to the family of string kernels, it is widely used to discover the patterns of biological sequences [15, 53, 54]. In the context of sequence analysis, the oligo kernel

is designed to work with oligomers occuring in sequences such as DNA and protein. They can be varying lengths but general tendency is to keep the length of the oligomers short. The DNA oligomers with length 2, for instance, consist of all possible 2-length-monomer combinations of the DNA alphabet $\mathscr{A}$, where $\mathscr{A} = \{A, T, C, G\}$. To be more specific, the DNA oligomers with length 2 form a set $\mathscr{A}^2$ where $\mathscr{A}^2 = \{AA, AT, AC, AG, TA, TT, TC,$
$TG, CA, CT, CC, CG, GA, GT, GC, GG\}$. In general, all possible combinations of K-length-monomers in an alphabet $\mathscr{A}$ of a sequence type are called K-mers and they form a set $\mathscr{A}^K$. These K-mers are utilized by the oligo kernel to determine the similarity between sequences. For each K-mer $\omega \in \mathscr{A}^K$ occurring in a sequence S, the corresponding oligo function μ is calculated as:

$$\mu_\omega(x) = \sum_{p \in S_\omega} \exp(-\frac{1}{2\sigma^2}(x-p)^2) \tag{I.1}$$

where S_ω is the set of occurrences of K-mer ω in sequence S, σ is the positional tolerance parameter for inexact matches. Based on the oligo function of each K-mer, the mapping function Φ in the oligo kernel is defined as follows:

$$\Phi^K(s) = [\mu_{\omega_1}, \mu_{\omega_2}, \cdots, \mu_{\omega_n}]$$

where s is the sequence and n is the size of the set $\mathscr{A}^K$. Even though this representation is suitable for visualization and interpretation, we need to have the kernel function. As stated in [15], the kernel function is calculated as follows:

$$k(s_i, s_j) = \sqrt{\pi}\sigma \sum_{\omega \in \mathscr{A}^K} \sum_{p \in S^i_\omega} \sum_{q \in S^j_\omega} \exp(-\frac{1}{4\sigma^2}(p-q)^2) \tag{I.2}$$

where s_i and s_j are the sequences, σ is the positional tolerance parameter, $\mathscr{A}^K$ is the set of all possible K-mers of monomers in alphabet $\mathscr{A}$, S^i_ω and S^j_ω are the set of occurrences of ω in sequence s_i and sequence s_j, respectively.

Unlike other similar approaches, the oligo kernel can be adjusted in a way that positional inexact matches of K-mers would also contribute to the similarity of sequences. The degree of this positional independence can be manipulated by the parameter σ. If σ is set close to 0, then only the exact matches of K-mers would contribute to the similarity. On the other hand, if σ is set to ∞, then there would be no importance of positions of K-mers.

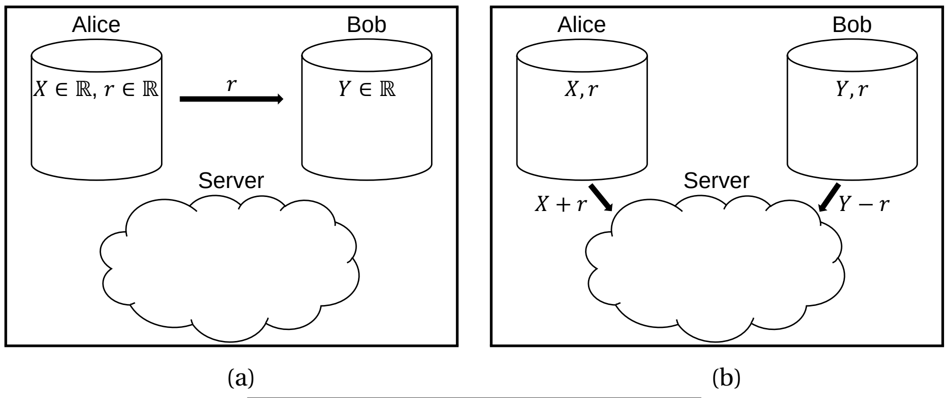
Based on Equation I.2, the calculation of the oligo kernel requires the differences of the positions of K-mers in both sequences. In our framework, we address the required operation and enable the computation of the differences privately.

I.3.2 Radial Basis Function Kernel

The radial basis function (RBF) is one of the most popular kernel functions in kernel learning algorithms [55]. It is commonly used in many different areas [56, 57, 58]. For samples $x, y \in \mathbb{R}^n$, the RBF kernel can be formulated solely based on the dot product of samples as follows:

$$K(x, y) = \exp\left(-\frac{\|x \cdot x - 2x \cdot y + y \cdot y\|^2}{2\sigma^2}\right) \tag{I.3}$$

where "·" represents the dot product of vectors and σ is the parameter that adjusts the similarity level. As shown in Equation I.3, the calculation of the RBF kernel between the samples x and y can be done by the dot product, which consists of the element-wise multiplication and summing up the results of these multiplications. In our framework, we allow the private computation of the RBF kernel by enabling the element-wise multiplication of the vectors.



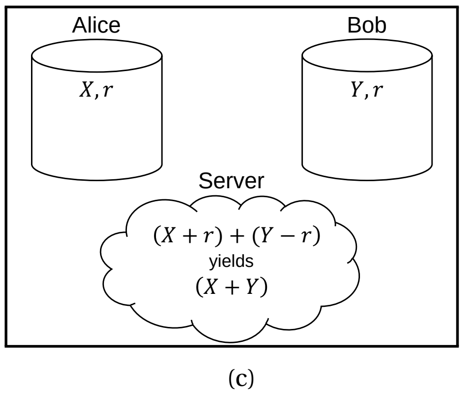


Figure I.1: In this setting, Alice and Bob are the input-parties and the server is the function-party. **(a)** Alice creates a uniformly chosen random value r. She shares it with Bob. **(b)** Once Bob receives the random value from Alice, he computes $Y - r$ and shares it with the server. Meanwhile, Alice computes $X + r$ and sends it to the server. **(c)** When the server receives the components of encoding, it calculates $(X + r) + (Y - r)$ to decode the result of the addition of the input values of Alice and Bob. At the end, the server obtains $X + Y$ without learning neither X nor Y. Similarly, none of the input-parties learns about the input value of the other input-party.

I.3.3 Randomized Encoding

In the cryptography literature, randomized encoding is proposed to compute a function $f(x)$ by a randomized function $\hat{f}(x; r)$, where r is a uniformly chosen random value, without revealing the input value x [59, 60]. The formal definition of the randomized encoding is as follows:

Definition 1 (Randomized Encoding [24])**.** Let us define a function $f : X \to Y$. There exists a function $\hat{f} : X \times R \to Z$ which is a δ-correct, (t, ϵ)-private randomized encoding of f if randomized algorithms, decoder Dec and simulator Sim, can be defined and the followings hold for these algorithms:

- (δ-correctness) $\forall x \in X$:

$$\Pr_{r \leftarrow R}[\mathsf{Dec}(\hat{f}(x; r)) \neq f(x)] \leq \delta.$$

- ((t,ϵ)-privacy) $\forall x \in X$ and any circuit $\mathscr{A}$ of size t:

$$\left|\Pr[\mathscr{A}(\mathsf{Sim}(f(x))) = 1] - \Pr_{r \leftarrow R}[\mathscr{A}(\hat{f}(x;r)) = 1]\right| \leq \epsilon.$$

where Dec decodes the given encoding and Sim simulates the encoding such that simulation and real encoding are indistinguishable.

Besides the formal definition of the randomized encoding, they also proposed two perfect decomposable and affine randomized encodings (DARE) for addition and multiplication of two values. In order for a randomized encoding to be affine and decomposable, each component of randomized encoding should be an affine function over the set that the function is defined and they should depend on only a single input value and a varying number of random values.

Definition 2 (Perfect RE for Addition [24])**.** Let us define a function $f(x_1, x_2) = x_1 + x_2$ over some finite ring R. This addition function can be perfectly encoded by the following DARE:

$$\hat{f}(x_1, x_2; r) = (x_1 + r, x_2 - r)$$

where r is a uniformly chosen random value. The encoding can be decoded by summing up the components of the encoding, and one can simulate the function by sampling two random values whose sum is y.

Definition 3 (Perfect RE for Multiplication [24])**.** Let us define a function $f(x_1, x_2) = x_1 \cdot x_2$ over a ring R. This multiplication function can be perfectly encoded by the following DARE:

$$\hat{f}(x_1, x_2; r_1, r_2, r_3) = (x_1 + r_1, x_2 + r_2, r_2 x_1 + r_3, r_1 x_2 + r_1 r_2 - r_3)$$

where r_1, r_2 and r_3 are uniformly chosen random values. Given the encoding (c_1, c_2, c_3, c_4), we can recover $f(x_1, x_2)$ by computing $c_1 \cdot c_2 - c_3 - c_4$. The simulator $\mathsf{Sim}(y; c_1, c_2, c_3) := (c_1, c_2, c_3, c_1 c_2 - y - c_3)$ perfectly simulates $\hat{f}$.

Randomized encoding preserves the privacy of the data by randomizing the input values and creating components by these values. At the end, it only allows the computation of the desired output and nothing else about the input values is revealed. Compared to the other methods in the literature such as homomorphic encryption and secure multi-party computation that result in a high overhead due to the use of computationally expensive cryptographic tools, the randomized encoding is faster and more efficient.

In the scenario where we have two input-parties and one function-party, the randomized encoding is also applicable. The randomized encoding of addition can be adapted to calculate the summation of two input values belonging to two different input-parties. The process is depicted in Figure I.1. The computation of the differences of two values can be done with the same encoding after multiplying the corresponding input value by -1. Similarly, one can employ the randomized encoding of multiplication in a function party in order to compute the multiplication of two input values owned by two distinct input-parties. The steps are demonstrated in Figure I.2. One can use the randomized encoding of multiplication to compute the division by simply using the reciprocal of the corresponding input value. It is worth to note that we assume that the value at the divisor should be non-zero.

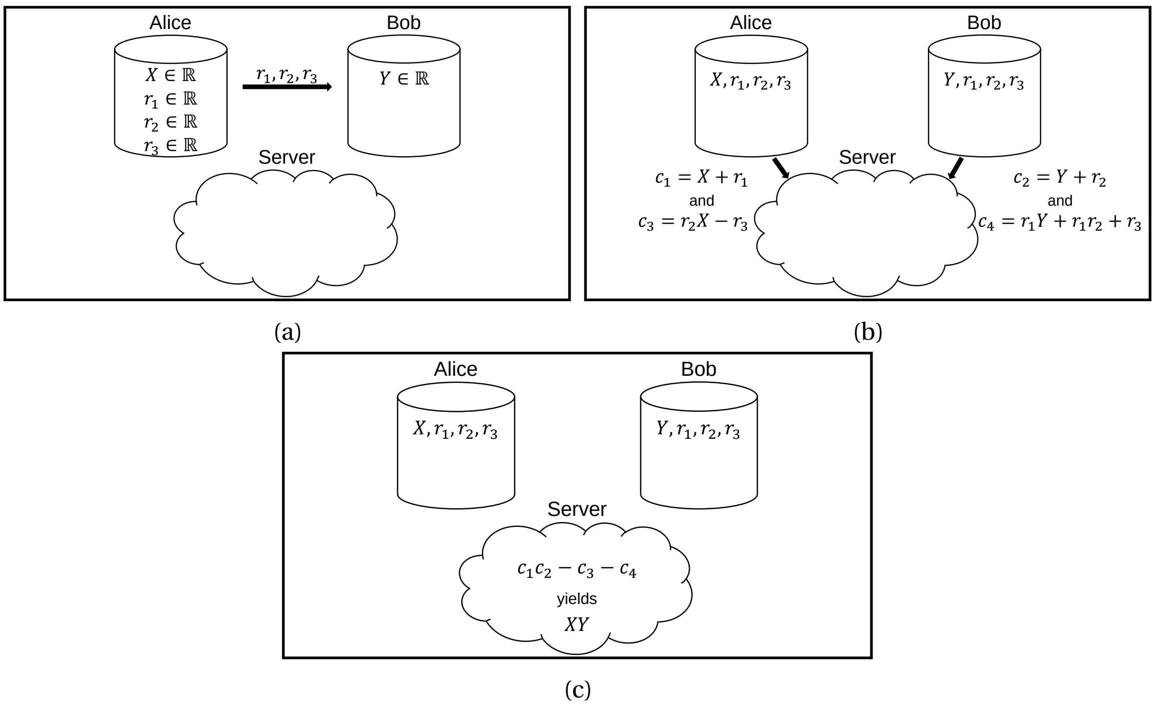
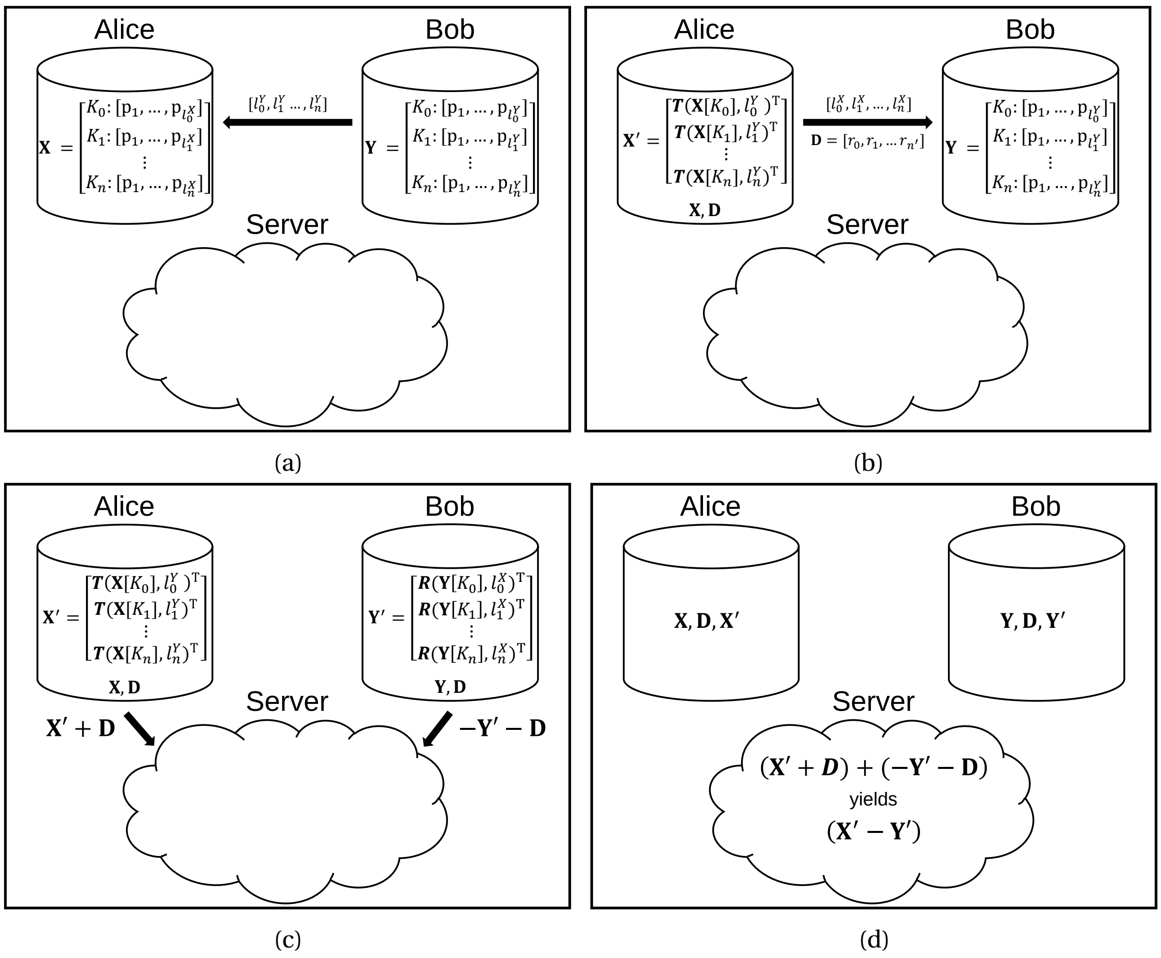


Figure I.2: In this setting, Alice and Bob are the input-parties and the server is the function-party. **(a)** Alice creates three uniformly chosen random values and shares them with Bob. **(b)** Once Bob receives the random values, he computes $c_2 = Y + r_2$ and $c_4 = r_1Y + r_1r_2 + r_3$. In the meantime, Alice computes $c_1 = X + r_1$ and $c_3 = r_2X - r_3$. They send c_1,c_2,c_3 and c_4 to the server. **(c)** When the server receives the components of the encoding, it calculates $c_1c_2 - c_3 - c_4$ to decode the result of the multiplication of the input values of Alice and Bob. At the end, the server obtains XY without learning neither X nor Y. Similarly, none of the input-parties learns about the input value of the other input-party.

I.4 Our Framework

In this paper, we propose a framework utilizing randomized encoding to support the computation of four basic arithmetic operations over the vectors of two input-parties in a function-party privately. In this application, we focus on the computation of differences and multiplications of these vectors. Our framework allows the server to learn only the intended outcome, either the element-wise differences or multiplications of the vectors, and nothing else about these vectors. Similarly, an input-party learns neither the input vector of the other input-party nor the result of the computation. Among the features of the framework, we use the element-wise differences of the vectors to compute the oligo kernel on the data owned by the input-parties. Additionally, we employ element-wise multiplication feature in order to compute the RBF kernel on the same setting. However; the framework is not limited to these methods. One can use the framework to compute a function which requires one of the features of the framework.



Alice
Bob
Server
$[l_0^Y, l_1^Y \dots, l_n^Y]$
$\mathbf{X} = \begin{bmatrix} K_0: [p_1, \dots, p_{l_0^X}] \\ K_1: [p_1, \dots, p_{l_1^X}] \\ \vdots \\ K_n: [p_1, \dots, p_{l_n^X}] \end{bmatrix}$
$\mathbf{Y} = \begin{bmatrix} K_0: [p_1, \dots, p_{l_0^Y}] \\ K_1: [p_1, \dots, p_{l_1^Y}] \\ \vdots \\ K_n: [p_1, \dots, p_{l_n^Y}] \end{bmatrix}$
(a)
$[l_0^X, l_1^X, \dots, l_n^X]$
$\mathbf{D} = [r_0, r_1, \dots r_{n'}]$
$\mathbf{X}' = \begin{bmatrix} \boldsymbol{T}(\mathbf{X}[K_0], l_0^Y)^\mathrm{T} \\ \boldsymbol{T}(\mathbf{X}[K_1], l_1^Y)^\mathrm{T} \\ \vdots \\ \boldsymbol{T}(\mathbf{X}[K_n], l_n^Y)^\mathrm{T} \end{bmatrix}$
$\mathbf{X}, \mathbf{D}$
(b)
$\mathbf{Y}' = \begin{bmatrix} \boldsymbol{R}(\mathbf{Y}[K_0], l_0^X)^\mathrm{T} \\ \boldsymbol{R}(\mathbf{Y}[K_1], l_1^X)^\mathrm{T} \\ \vdots \\ \boldsymbol{R}(\mathbf{Y}[K_n], l_n^X)^\mathrm{T} \end{bmatrix}$
$\mathbf{Y}, \mathbf{D}$
$\mathbf{X}' + \mathbf{D}$
$-\mathbf{Y}' - \mathbf{D}$
(c)
$\mathbf{X}, \mathbf{D}, \mathbf{X}'$
$\mathbf{Y}, \mathbf{D}, \mathbf{Y}'$
$(\mathbf{X}' + \boldsymbol{D}) + (-\mathbf{Y}' - \mathbf{D})$
yields
$(\mathbf{X}' - \mathbf{Y}')$
(d)

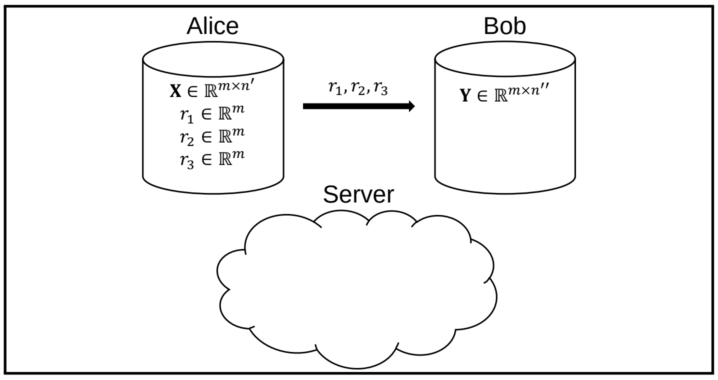
Figure I.3: In this figure, $\mathbf{K_i}$ represents the vector containing the positions of the occurrences of the i-th oligomer in the corresponding sequences. In those vectors, p_j of $\mathbf{K_i}$ represents the position of the j-th occurrence of the i-th oligomer. **(a)** Both input-parties find the positions of the oligomers in their sequences and insert a varying number of dummy positions into randomly chosen oligomers. Then, Bob shares the length of $\mathbf{K_j}$ $\forall j \in \{0, \cdots, n\}$ with Alice. **(b)** Afterwards, Alice creates the vector $X' \in \mathbb{R}^{n'}$ by applying the function $\mathbf{T}$ along with the vector $D \in \mathbb{R}^{n'}$ for encoding. Then, she sends D and the length of $\mathbf{K_i}$ $\forall i \in \{0, \cdots, n\}$ to Bob. **(c)** Bob creates the vector $Y' \in \mathbb{R}^{n'}$ by applying the function R. Next, they calculate $(-Y' - D) \in \mathbb{R}^{n'}$ and $(X' + D) \in \mathbb{R}^{n'}$, respectively, and share them with the server. **(d)** At the end, the server computes the element-wise summation of the given vectors to obtain all possible pairwise differences of the positions for each oligomer. Once the server prunes the entries involving dummy values, it can compute the oligo kernel function.

I.4.1 Computation of Oligo Kernel

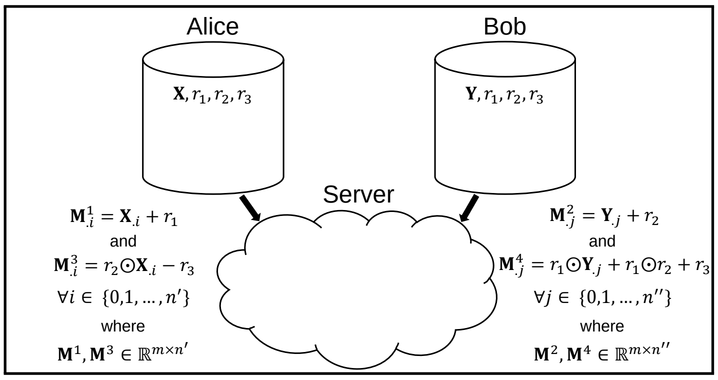
We utilize the randomized encoding of addition in order to compute the oligo kernel. Since the calculation of difference is required to compute the oligo kernel, we adapt the encoding of addition by taking the negative of the input value of one of the input-parties. The process of the computation of the oligo kernel is depicted in Figure I.3. Let us assume that Alice and Bob have hashmap-like input vectors X and Y where $X, Y \in \mathbb{R}^n$ and n is the number of different oligomers. These vectors contain K_i representing the vector of positions of i-th oligomer in the sequence of the corresponding input-party for $i \in \{0, 1, ..., n\}$. The length of the vector K_i in X is denoted by l_i^X. It is worth to note that we have non-empty vectors for all oligomers in Figure I.3 due to the illustration purposes. In fact, K_i can be any length vector including empty vector, that is $K_i \in \{\emptyset, \mathbb{R}^1, \mathbb{R}^2, \cdots, \mathbb{R}^M\}$ $\forall i \in \{0, \cdots, n\}$ where M is the maximum possible number of oligomer that could occur in a specific length of sequences. First of all, the input-parties find the positions of oligomers in their sequences. Before Bob shares the length of vectors K_j $\forall j \in \{0, 1, ..., n\}$ with Alice, that is the vector $[l_0^Y, l_1^Y, \cdots, l_n^Y]$, Bob inserts varying number of dummy positions into randomly chosen oligomers where the position value is smaller than possible. In our case, it is -10^9. Similarly, Alice inserts some number of dummy positions into randomly chosen oligomers where the position is too large. For Alice, this value is 10^9. After receiving the number of occurrences of oligomers in the sequence of Bob, Alice applies the function $\boldsymbol{T}$, which repeats the vector as a whole for the specified number of times, over her oligomers. As an example, $\boldsymbol{T}([2,5],3)$ yields $A = [2,5,2,5,2,5]$. Once Alice has the transpose of these repeated vectors, she concatenates them to create the vector $X' \in \mathbb{R}^{n'}$ where n' is the length of the vector after concatenation. Alice also creates the vector $D \in \mathbb{R}^{n'}$ having uniformly chosen random values for each entry of X'. Afterwards, Alice shares the vector D and the number of occurrences of her oligomers, that is the vector $[l_0^X, l_1^X, \cdots, l_n^X]$. It is important to note that all oligomers exist and they are common among input-parties due to the illustration purpose. However, in case of missing oligomers, Alice can work on and send only the common oligomers among the input-parties in order to reduce the communication cost. Once Bob gets the number of occurrences of oligomers in Alice and the vector D, he applies the function $\boldsymbol{R}$, which repeats each entry of the given vector for the specified number of times, over his oligomer vectors. As a clarification, $\boldsymbol{R}([2,5],3)$, for instances, yields $A = [2,2,2,5,5,5]$. After Bob has the transpose of these repeated vectors, he concatenates them to create the vector $Y' \in \mathbb{R}^{n'}$ and calculates $(-Y' - D)$. In the meantime, Alice computes $(X' + D)$. Then, the input-parties share these vectors with the server. After the server receives the components of the encoding, it computes the summation of these components in order to obtain all possible pairwise differences of the positions for each oligomer in the sequences of Alice and Bob. At this point, the obtained vector contains the entries which are the results of the operation involving dummy values in addition to the actual entries. Since the server knows the artificial position values of Alice and Bob, it can detect whether a result is valid or not. If the absolute of a result is larger than or close to the artificial position value, then the server ignores it in the computation. At the end, the purified vector is utilized to compute the oligo kernel function over two sequences via Equation I.2. In order to compute the kernel matrix via the oligo kernel for multiple sequences in each input-party, we repeat the same process for all pairs of sequences from Alice and Bob. The differences among the sequences in the same party can be computed by that party and shared with the server to complete the kernel matrix. It is worth to note that sharing the number of occurrences of oligomers can be done once for all in order to reduce the communication cost.

I.4.2 Computation of RBF Kernel

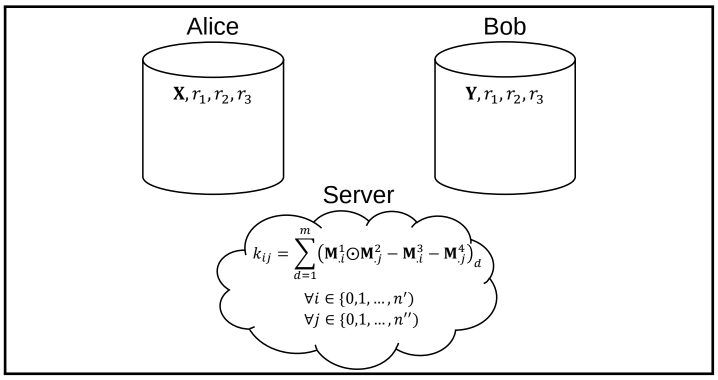
We employ the randomized encoding for multiplication to compute the RBF kernel over the data of the input-parties. The first step of the computation is to calculate the element-wise multiplication of the vectors from different input-parties. We utilize the randomized encoding to overcome this problem. The process is demonstrated in Figure I.4. In the computation, let us assume that Alice and Bob have input matrices X and Y, respectively, whose columns represent samples and $A_{.i}$ shows the vector at the i-th column for any matrix A. In order to create the components of the



(a)


Alice
Bob
$\mathbf{X}, r_1, r_2, r_3$
$\mathbf{Y}, r_1, r_2, r_3$
Server
$\mathbf{M}^1_{.i} = \mathbf{X}_{.i} + r_1$
and
$\mathbf{M}^3_{.i} = r_2 \odot \mathbf{X}_{.i} - r_3$
$\forall i \in \{0,1,\ldots,n'\}$
where
$\mathbf{M}^1, \mathbf{M}^3 \in \mathbb{R}^{m \times n'}$
$\mathbf{M}^2_{.j} = \mathbf{Y}_{.j} + r_2$
and
$\mathbf{M}^4_{.j} = r_1 \odot \mathbf{Y}_{.j} + r_1 \odot r_2 + r_3$
$\forall j \in \{0,1,\ldots,n''\}$
where
$\mathbf{M}^2, \mathbf{M}^4 \in \mathbb{R}^{m \times n''}$

(b)


Alice
Bob
$\mathbf{X}, r_1, r_2, r_3$
$\mathbf{Y}, r_1, r_2, r_3$
Server
$k_{ij} = \sum_{d=1}^{m} \left(\mathbf{M}^1_{.i} \odot \mathbf{M}^2_{.j} - \mathbf{M}^3_{.i} - \mathbf{M}^4_{.j}\right)_d$
$\forall i \in \{0,1,\ldots,n')$
$\forall j \in \{0,1,\ldots,n'')$

(c)

Figure I.4: **(a)** In order to create the components of the encoding, Alice creates three vectors with uniformly chosen random values of the same length with a sample in the input vector X. Then, she sends these vectors to Bob. **(b)** When Bob receives the random values, he computes the random components M^2 and M^4. Meanwhile, Alice computes her shares of encoding, namely M^1 and M^3. Eventually, they send these components of the randomized encoding to the server. **(c)** Once the server receives the components, it computes the dot products between the i-th sample of Alice and the j-th sample of Bob by summing up the entries of the vector $(M^1_{.i} \odot M^2_{.j} - M^3_{.i} - M^4_{.j})$. The server repeats this process for all pairs of samples of Alice and Bob.

encoding, Alice creates three vectors with uniformly chosen random values of the same length with a sample in the input vector X. Afterwards, Alice shares these vectors with Bob. When Bob receives the random values, he computes the random components M^2 and M^4. Meanwhile, Alice computes her shares of encoding, namely M^1 and M^3. Eventually, they send these components of randomized encoding to the server. Once the server receives the components, it computes the dot products between the i-th sample of Alice and the j-th sample of Bob by summing up the entries of the vector $(M^1_{.i} \odot M^2_{.j} - M^3_{.i} - M^4_{.j})$ where "$\odot$" represents the Hadamard product. The server repeats this process for all pairs of samples of Alice and Bob. In order to complete the *gram matrix* which indicates the inner product of the vectors, the input-parties compute the dot product among their own samples and send the resulting matrices to the server. Then, the server employs Equation II.3.3, in which the RBF kernel computation via the inner product is shown, to compute the RBF kernel matrix.

I.5 Security Analysis

In this section, we give the threat model that we used to assess the security of the framework. Based on the threat model, we evaluated the security of the oligo kernel and the RBF kernel calculations.

I.5.1 Threat Model

We use the semi-honest adversary model in which a computationally bounded adversary that is not allowed to deviate from the protocol description attempts to obtain a valuable information from the messages sent during the execution of the protocol. In other words, all parties follow the protocol specification, but the parties may try to obtain additional information about the private input values of other parties based on their views on the execution of the protocol. In our framework, the input-parties try to keep their data private so, they can be assumed to be trusted parties not to actively cheat. The function-party wants to perform arithmetic operations on the data of the input-parties. It is expected that the function-party actively misbehaves in order to obtain the data of the input-parties. However, the function-party does not send messages to the input-parties in our framework. Therefore, it has to make a coalition with one of the input-parties in order to obtain the data of the other input-party.

I.5.2 Oligo Kernel Computation

Lemma 1. Let $\mathscr{A}$ be a semi-honest adversary. The advantage of $\mathscr{A}$ of obtaining the positions of oligomers by analyzing the number of occurrences of oligomers is negligible .

Proof. Alice and Bob share the number of occurrences of oligomers $[l^X_0, l^X_1, \cdots, l^X_n]$ and $[l^Y_0, l^Y_1, \cdots, l^Y_n]$ to each other. $\mathscr{A}$ can try to reconstruct Alice's sequence and Bob's sequence by using $[l^X_0, l^X_1, \cdots, l^X_n]$ and $[l^Y_0, l^Y_1, \cdots, l^Y_n]$, respectively. We assume that $\mathscr{A}$ can obtain some possible position information of the oligomers from all possible sequences that she can construct. The success rate of the attack increases with the increase in the size of oligomers, that is the value of K. Alice and Bob add a varying number of dummy positions for randomly chosen oligomers. This method increases the number of possible sequences that $\mathscr{A}$ can construct, and thus reduce the success rate of this attack to negligible levels. $\square$

Theorem 2. The computation of the oligo kernel described in Section I.4.1 is secure in the presence of a semi-honest adversary $\mathscr{A}$.

Proof. In the computation of the oligo kernel, Alice and Bob compute $(X' + D)$ and $(-Y' - D)$, respectively, where X' and Y' are vectors of their secret values and D is the vector of random values. The calculation is directly based on the randomized encoding for addition. The only difference is that it is made over multiple values. Alice and Bob share the number of occurrences of oligomers $L^X = [l_0^X, l_1^X, \cdots, l_n^X]$ and $L^Y = [l_0^Y, l_1^Y, \cdots, l_n^Y]$ to each other during the calculation. We assume that there is a semi-honest adversary $\mathscr{A}$ that can obtain X' or Y' with non-negligible probability. $\mathscr{A}$ can use L^X or L^Y in order to obtain X' or Y', respectively. However, $\mathscr{A}$ cannot obtain X' or Y' from L^X or L^Y, respectively (Lemma 1). $\mathscr{A}$ has to decode X' and Y' from messages $(X' + D)$ and $(-Y' - D)$. However, this contradicts with the privacy property of the perfect randomized encoding of addition function (Definition 2). $\square$

I.5.3 RBF Kernel Computation

Theorem 3. The computation of RBF kernel described in Section I.4.2 is secure in the presence of semi-honest adversary $\mathscr{A}$.

Proof. In the computation of the RBF kernel, Alice and Bob compute $\{(X_{.i} + r_1), (r_2 \odot X_{.i} - r_3)\}$ and $\{(Y_{.j} + r_2), (r_1 \odot Y_{.j} + r_1 \odot r_2 + r_3)\}$, respectively, where $X_{.i}$ and $Y_{.j}$ are the column vectors of their secret values, and r_1, r_2 and r_3 are the vectors of random values. The calculation is directly based on the randomized encoding of multiplication. The only difference is that it is made over multiple values. We assume that there is a polynomial-time adversary $\mathscr{A}$ that can decode $X_{.i}$ and $Y_{.j}$ from messages $(X_{.i} + r_1), (r_2 \odot X_{.i} + r_3), (Y_{.j} + r_2)$ and $(r_1 \odot Y_{.j} + r_1 \odot r_2 + r_3)$. This contradicts with the privacy property of perfect randomized encoding of the multiplication function (Definition 3). $\square$

I.6 Dataset

To show the benefit of privacy-preserving machine learning, we chose a genomic data set with clinical relevance in precision medicine. It comprises V3 loop sequences of HIV together with the coreceptor usage as the label. Coreceptor usage determines, how the viruses are entering the human cells. Since the most common variants can only use the human CCR5 coreceptor, which can be blocked by a drug, it is very important to determine the coreceptor usage of the viral population before prescribing those antiretroviral drugs [61]. Over recent years several successful applications based on predicting the phenotype from genetic data have been introduced [62]. We downloaded publically available data from the Los Alamos National Laboratory (LANL) HIV Sequence Database at http://www.hiv.lanl.gov/. We chose all amino acid sequences with coreceptor usage information and then classified the data into two classes (CCR5 only versus OTHER). We had 642 and 124 sequences in each class, respectively. The sequences were aligned with the HIVAlign tool from the LANL website with standard options. At the end, we obtained 766 sequences with 44 characters each.

I.7 Results and Discussion

We utilized the framework to calculate the oligo kernel and the RBF kernel. We applied these kernels on V3 loop sequences of HIV to predict the coreceptor usage by employing support vector machines [16]. We shared the labels of the samples with the server in plaintext domain since it does not reveal any additional information. We determined the best parameters according to F1-score by 5-fold cross-validation. In this process, we evaluated different values for the kernel parameters, which are the positional tolerance parameter $\sigma \in \{2^{-5}, 2^{-4}, \cdots, 2^{10}\}$ of the oligo kernel and the similarity adjustment parameter $\sigma \in \{2^{-5}, 2^{-4}, \cdots, 2^{10}\}$ of the RBF kernel. Similarly, we evaluated different values for the SVM parameters, which are the weight $w_1 \in \{2^0, 2^1, \cdots, 2^5\}$ of the minority class *OTHER* and the misclassification penalty parameter $C \in \{2^{-5}, 2^{-4}, \cdots, 2^{10}\}$. We repeated the parameter optimization step 10 times with different random folds and conducted the corresponding experiments with each set of optimal parameters separately. We evaluated the results of the experiments by employing F1-score and area under receiver operating characteristic (AUROC) curve. In the experiments, we selected the random values from the range $[1, 100]$.

In the experiments with the proposed framework, we employed three processes. Two of them are the input-parties and the last one is the function-party. We split the data into input-parties equally and used around 20% of the data of each input-party as test data. We conducted the experiments on a server having 512 GB memory, Intel Xeon E5-2650 processor and 64-bit operating system. We let the parties communicate with each other over TCP sockets and assumed that the communication is secure. We implemented the framework in Python. This holds also for the key switching approach [31] to which we compare our approach.

I.7.1 Oligo Kernel Experiments

In the oligo kernel experiments, we utilized the proposed framework to calculate the kernel function without sacrificing the privacy of the sequence data. Since we were not able to find a study which aimed for a similar scenario and utilized the oligo kernel, we compared our approach to the non-private oligo kernel computation in which we simply ran the oligo kernel in a single party where the whole data is available. The execution time of both private scheme (PP) and non-private scheme (NP) is shown in Figure I.5. Based on the figure, it can be stated that the privacy setup in the framework does not put too much extra burden on the execution time of the computation in PP. The private computation of the oligo kernel function is still done within a reasonable time. Moreover, the actual time required to obtain the results in PP is around one third of the given total execution time of PP since we utilized three processes, which are two processes for input-parties and one process for the function-party, to compute the function. Figure I.5 displays the total execution time of these processes.

The execution time for the experiments of both schemes with larger K-mers tend to decrease with larger K values due to the decrease in the number of occurrences of K-mers in the sequences. The number of possible occurrences of K-mers in a sequence of length l is $l - K - 1$ and it gets lower by the increase of K. Furthermore, since the sequences that we employ have gap characters due to the alignment, the number of K-mers occurring in the sequences decreases parallel to the increase of K in our experiments considering that we do not allow gapped K-mers. Both approaches produced the same results in terms of F1-score and AUROC, and Figure I.6 demonstrates these results. Having the same results indicates that our framework is able to calculate the exact differences without

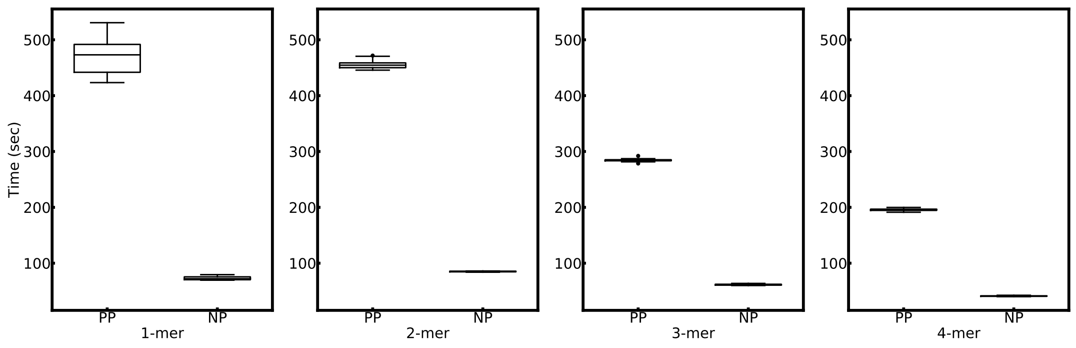


Figure I.5: We compare the execution time of our framework (PP) to the non-private scheme (NP) in the oligo kernel experiments. It shows that our framework is promising for real-life applications.

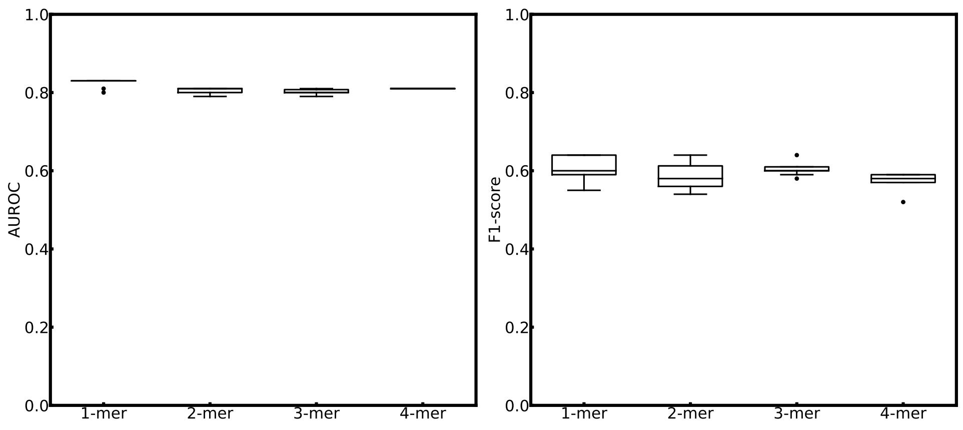


Figure I.6: In each K-mer, we have 10 different experiments. The experiments of both the private scheme and the non-private scheme with the oligo kernel yield the same results. Therefore, we give a single plot for each type of evaluation metric to display the results of both schemes. The results are better towards small K values due to the size of the alphabet of the protein sequences.

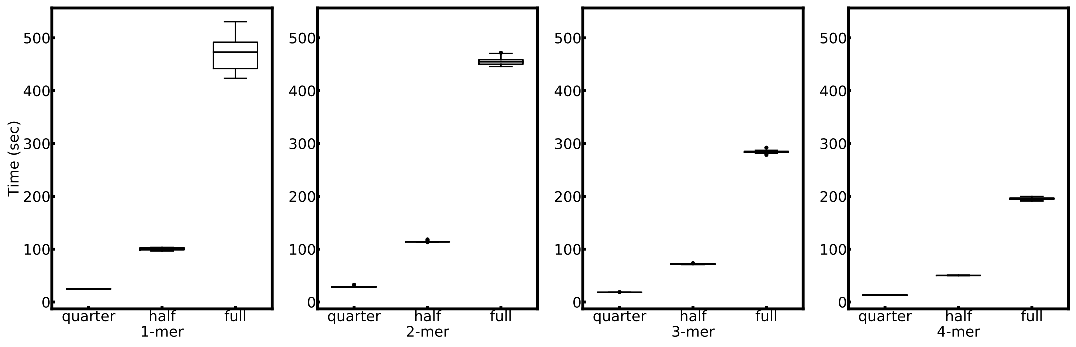
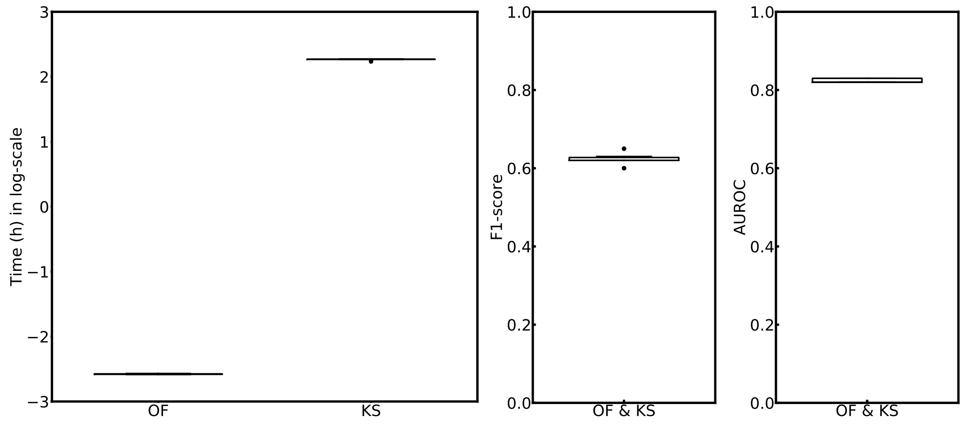


Figure I.7: The execution time of our approach in the oligo kernel experiment for a varying size of the dataset is depicted for 10 repetitions for each size and different K-mer lengths. It scales almost quadratically with the size of dataset for all K values.

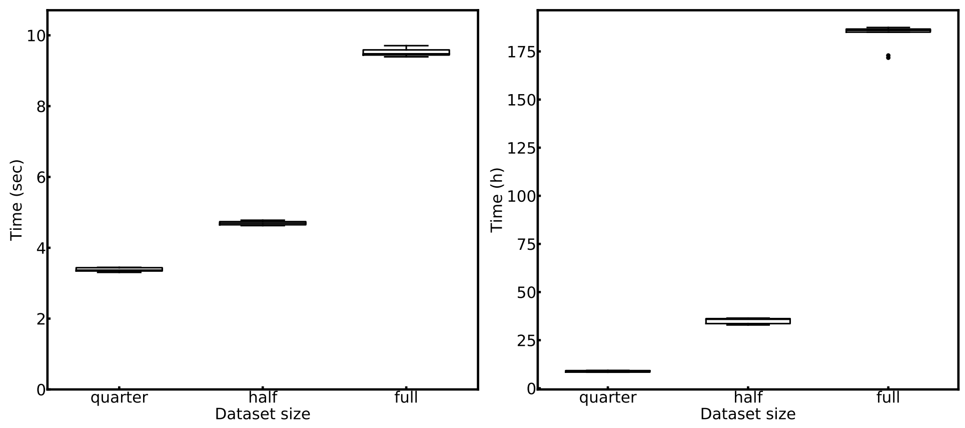
sacrificing the privacy of the data.

We further conducted experiments on our proposed framework by using datasets of varying sizes to demonstrate the scalability of the framework. We used a quarter, a half and the full of the dataset. The execution time for these experiments are depicted in Figure I.7. The complexity of the framework grows almost quadratically with the size of dataset.



(a) The execution time (b) F-measure and AUROC

Figure I.8: **(a)** In the RBF kernel experiment, we compare the execution time of our framework (OF) to the key switching approach (KS) in log-scale. In each approach, we have 10 repetitions of the experiments. **(b)** Both OF and KS experiments yielded the same results in terms of F1-score and AUROC. The RBF kernel yields comparable or even better results than the oligo kernel.



(a) (b)

Figure I.9: **(a)** The execution time of our approach in the RBF kernel experiment for varying size of the dataset is shown. It scales almost quadratically with the size of dataset. **(b)** Similarly, the execution time of the key switching approach to compute the RBF kernel scales quadratically with the size of dataset. Note that the units are seconds and hours, respectively, in the figures.

I.7.2 RBF Kernel Experiments

As an alternative approach, we employed the RBF kernel to predict the coreceptor usage of HIV. In order to prepare the data for the RBF kernel, we encoded the sequences with one-hot-encoding such that each amino acid in a sequence was represented by 21 bits in which only one of those bits is 1 and the rest are 0. We utilized the formula given in Equation II.3.3 to calculate the RBF kernel on the samples of input-parties, which requires us to perform the element-wise multiplication of the feature vectors of these samples. In order to compare the performance of our proposed framework to existing approaches, we used key switching approach [31] which utilized the idea in [32, 63]. We selected the length of the secret key in the key switching approach as 10 which was the length of the secret key in their experiments. In both experiments, the dot product among the samples in the same input-party is calculated directly in that input-party and the result is sent to the server. To compute the dot product of the samples from different input-parties, we utilized our proposed approach and the key switching approach separately. Both approaches gave the same results for the same set of parameters. F1-score and AUROC of these experiments are shown in Figure I.8b. Based on the results, the RBF kernel can be considered as a competitive alternative to the oligo kernel for this prediction scenario.

The execution time of both approaches are shown in Figure I.8a. It indicates that our approach is way faster than the key switching approach. The underlying reason for such a difference is that the number of features of the samples affects the execution time of the key switching approach drastically [31]. In our case, the length of the feature vectors is $21 * 44$, that is 924. Whereas, our approach can handle high dimensional vectors efficiently compared to the key switching approach and it does not involve any computationally expensive encryption. Moreover, we can conclude that our approach is more efficient than the computation of the dot product via HELib [64, 65] since integer vector homomorphic encryption is faster than HELib based dot product calculation [63].

I.8 Conclusion

Due to the necessity of data sharing in biomedical studies, privacy preservation becomes very essential. Especially in genetics, the protection of the sequence data of patients in the studies involving two sources requires the development of new privacy methods. In order to address this, we propose a framework utilizing randomized encoding in order to enable the computation of machine learning algorithms with a single basic arithmetic operation on the data from two sources. During the computation, none of the input value of these sources is revealed to neither of the other parties in the computation. Moreover, the result of the computation is not revealed to the sources. We demonstrate the performance of the framework on the coreceptor usage prediction problem of HIV by utilizing V3 loop sequences. The results of experiments of the oligo kernel show that our framework can yield the same results as with non-private scheme. The execution time analysis of oligo kernel experiments shows that our framework is promising for real-life applications. In the experiments of the RBF kernel, we show that our framework can compute the kernel function without losing information. Moreover, it is significantly more efficient than the key switching approach and the dot product computation via HELib consequently. In addition to the utilized machine learning algorithms, our framework can be used on any problem requiring one type of four basic arithmetic operations on data from two sources. If two different basic arithmetic operations are required, then the function-party would infer the input values of the input-parties in the current setting by utilizing the information coming from different operations. As a future work, the proposed framework can be improved in a way that it could perform more than a single type of basic arithmetic operation in the same computation without sacrificing the privacy of the input values. Additionally, one can extend the idea to cover more than two input-parties in the computation.

Acknowledgement

This study is supported by the DFG Cluster of Excellence “Machine Learning – New Perspectives for Science”, EXC 2064/1, project number 390727645. Furthermore, NP and MA acknowledge funding from the German Federal Ministry of Education and Research (BMBF) within the ‘Medical Informatics Initiative’ (DIFUTURE, reference number 01ZZ1804D).

II Privacy Preserving Gaze Estimation using Synthetic Images via a Randomized Encoding Based Framework

Efe Bozkir* Ali Burak Ünal* Mete Akgün Enkelejda Kasneci Nico Pfeifer

Abstract

Eye tracking is handled as one of the key technologies for applications that assess and evaluate human attention, behavior, and biometrics, especially using gaze, pupillary, and blink behaviors. One of the challenges with regard to the social acceptance of eye tracking technology is however the preserving of sensitive and personal information. To tackle this challenge, we employ a privacy-preserving framework based on randomized encoding to train a Support Vector Regression model using synthetic eye images privately to estimate the human gaze. During the computation, none of the parties learn about the data or the result that any other party has. Furthermore, the party that trains the model cannot reconstruct pupil, blinks or visual scanpath. The experimental results show that our privacy-preserving framework is capable of working in real-time, with the same accuracy as compared to non-private version and could be extended to other eye tracking related problems.

II.1 Introduction

Recent advances in the fields of Head-Mounted-Display (HMD) technology, computer graphics, augmented reality (AR), and eye tracking enabled numerous novel applications. One of the most natural and non-intrusive ways of interaction with HMDs or smart glasses is achieved by gaze-aware interfaces using eye tracking. However, it is possible to derive a lot of sensitive and personal information from eye tracking data such as intentions, behaviors, or fatigue since eyes are not fully controlled in a conscious way.

It has been shown that cognitive load [66, 67], visual attention [68], stress [69], task identification [70], skill level assessment and expertise [33, 71, 72], human activities [73, 74], biometric information and authentication [75, 76, 77, 78, 79], or personality traits [80] can be obtained using eye tracking data. Since highly sensitive information can be derived from eye tracking data, it is not surprising that HMDs or smart glasses have not been adopted by large communities yet. According to a recent survey [81], people agree to share their eye tracking data only when it is co-owned by a governmental health-agency or is used for research purposes. This indicates that people are hesitant about sharing their eye tracking data in commercial applications. Therefore, there is a likelihood that larger communities could adopt HMDs or smart glasses if privacy-preserving techniques are applied in the eye tracking applications. The reasons why privacy preserving schemes are needed for eye tracking are discussed in [82] extensively. However, until now, there are not many studies in privacy-preserving eye tracking. Recently, a method to detect privacy sensitive everyday situations [83], an approach to degrade iris authentication while keeping the gaze tracking utility in an acceptable accuracy [84], and differential privacy based techniques to protect personal information on heatmaps and eye movements [85, 81] are introduced. While differential privacy can be applied to eye tracking data for various tasks, it introduces additional noise on the data which causes decrease in the utility [85, 81], and it might lead to less accurate results in computer vision tasks, such as gaze estimation or activity recognition.

In light of the above, function-specific privacy models are required. In this work, we focus on the gaze estimation problem as a proof-of-concept by using synthetic data including eye landmarks and ground truth gaze vectors. However, the same privacy-preserving approach can be extended to any feature-based, eye tracking problem such as intention, fatigue, or activity detection, in HMD or unconstrained setups due to the demonstrated real-time working capabilities. In our study, the gaze estimation task is solved by using Support Vector Regression (SVR) models in a privacy-preserving manner by computing the dot product of eye landmark vectors to obtain the kernel matrix of the SVR for a scenario, where two parties have the eye landmark data, each of which we call *input-party*, and one *function-party* that trains a prediction model on the data of the input-parties. This scenario is relevant when the input-parties use eye tracking data to improve the accuracy of their models and do not share the data due to the privacy concerns. To this end, we utilize a framework employing randomized encoding [86]. In the computation, neither the eye images nor the extracted features are revealed to the function-party directly. Furthermore, the input-parties do not infer the raw eye tracking data or result of the computation. Eye images that are used for training and testing are rendered using UnityEyes [87] synthetically and 36 landmark-based features [88] are used. To the best of our knowledge, this is the first work that applies a privacy-preserving scheme based on function-specific privacy models on an eye tracking problem.

II.2 Threat Model

We assume that the input-parties are semi-honest (honest but curious) that are not allowed to deviate from the protocol description while they try to infer some valuable information about other parties' private inputs using their views of the protocol execution. We also assume that the function-party is malicious and the input-parties and the function-party do not collude.

II.3 Methodology

In this section, we discuss the data generation, randomized encoding, and privacy-preserving gaze estimation framework.

II.3.1 Data Generation

To train and evaluate the gaze estimator, we generate eye images and gaze vectors. As our work is a proof-of-concept and requires high amount of data, synthetic images from UnityEyes [87], which is based on the Unity3D, are used. *Camera parameters* and *Eye parameters* are chosen as $(0,0,0,0)$ (fixed camera) and $(0,0,30,30)$ (eyeball pose range parameters in degrees), respectively. $20,000$ images are rendered in *Fantastic* quality setting and 512×384 screen resolution. Then, processing and normalization pipeline from [88] is employed. In the end, we obtain 128×96 sized eye images, 18 eye landmarks including eight iris edge, eight eyelid, one iris center, and one iris-center-eyeball-center vector normalized according to Euclidean distance between eye corners, and gaze vectors using pitch and yaw angles. Final feature vectors consist of 36 elements. Figure II.1 shows an example illustration.

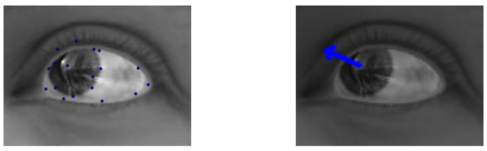

(a) Landmarks. (b) Gaze.

Figure II.1: Eye landmarks and gaze on a synthetic image.

II.3.2 Randomized Encoding

The utilized framework employs randomized encoding (RE) [59, 60] to compute the dot product of the landmark vectors. The dot product is needed to compute kernel matrix of the SVR which is later used for training the gaze estimator and validation of the framework.

In the randomized encoding, the computation of a function $f(x)$ is performed by a randomized function $\hat{f}(x;r)$ where x is the input value, which corresponds to eye landmarks in our setup, and r is the random value. The idea is to encode the original function by using random value(s) such that the combination of the components of the encoding reveals only the output of the original function. In the framework, the computation of the dot product is accomplished by utilizing the decomposable and affine randomized encoding (DARE) of addition and multiplication [24]. The encoding of multiplication is as follows.

Definition 4 (Perfect RE for Multiplication [24])**.** A multiplication function is defined as $f_m(x_1, x_2) = x_1 \cdot x_2$ over a ring R. One can perfectly encode the f_m by employing the DARE $\hat{f_m}(x_1, x_2; r_1, r_2, r_3)$:

$$\begin{aligned}\hat{f_m}(x_1, x_2; r_1, r_2, r_3) = (&x_1 + r_1, x_2 + r_2,\\ &r_2 x_1 + r_3, r_1 x_2 + r_1 r_2 - r_3),\end{aligned}$$

where r_1, r_2 and r_3 are uniformly chosen random values. The recovery of $f_m(x_1, x_2)$ can be accomplished by computing $c_1 \cdot c_2 - c_3 - c_4$ where $c_1 = x_1 + r_1$, $c_2 = x_2 + r_2$, $c_3 = r_2 x_1 + r_3$ and $c_4 = r_1 x_2 + r_1 r_2 - r_3$. The simulation of $\hat{f_m}$ can be done perfectly by the simulator $\mathsf{Sim}(y; a_1, a_2, a_3) := (a_1, a_2, a_3, a_1 a_2 - y - a_3)$ where a_1, a_2 a_3 are random values.

II.3.3 Framework

To perform the private gaze estimation task in our scenario, we inspire from the framework as in [86] due to its efficiency compared to other approaches in the literature. The framework is proposed to compute the addition or multiplication of the input values of two input-parties in the function-party by utilizing randomized encoding. We utilize the multiplication operation over the eye landmark vectors to compute the dot product of these vectors to obtain kernel matrix of the SVR in a privacy-preserving way.

We have two input-parties as Alice and Bob, having the eye landmark data as $X \in \mathbb{R}^{n_f \times n_a}$ and $Y \in \mathbb{R}^{n_f \times n_b}$ where n_a and n_b represent the number of samples in Alice and Bob, respectively, and n_f is the number of features. In addition to the input-parties, there exists a server that trains a model on the data of the input-parties. $A_{.j}$ for any matrix A represents the j-th column of the corresponding matrix and "$\odot$" represents the element-wise multiplication of the vectors. As a first

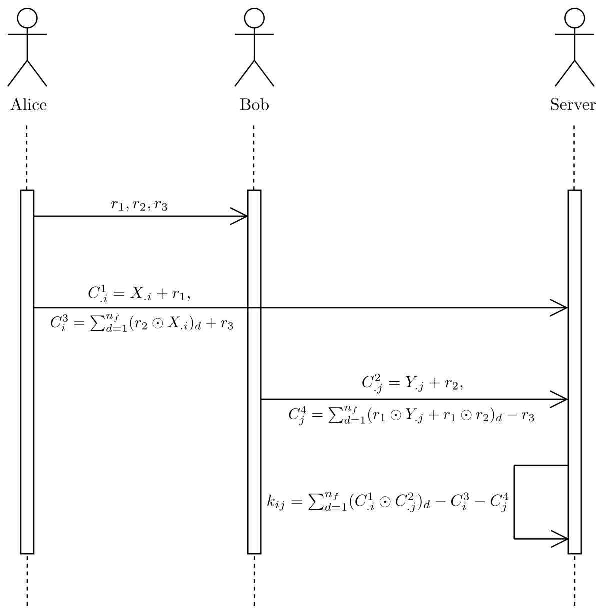

Alice
Bob
Server
r_1, r_2, r_3
$C^1_{.i} = X_{.i} + r_1,$
$C^3_i = \sum_{d=1}^{n_f} (r_2 \odot X_{.i})_d + r_3$
$C^2_{.j} = Y_{.j} + r_2,$
$C^4_j = \sum_{d=1}^{n_f} (r_1 \odot Y_{.j} + r_1 \odot r_2)_d - r_3$
$k_{ij} = \sum_{d=1}^{n_f} (C^1_{.i} \odot C^2_{.j})_d - C^3_i - C^4_j$

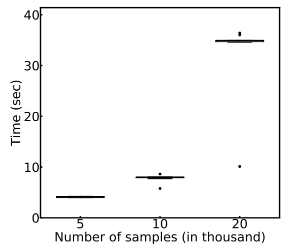
Figure II.2: Overall protocol execution.

step, Alice creates a uniformly chosen random value $r_3 \in \mathbb{R}$ and two vectors $r_1, r_2 \in \mathbb{R}^{n_f}$ with uniformly chosen random values, which are used to encode the element-wise multiplication of the vectors and shares them with Bob. Afterwards, Bob computes $C^2_{.j} = Y_{.j} + r_2$ and $C^4_j = \sum_{d=1}^{n_f} (r_1 \odot Y_{.j} + r_1 \odot r_2)_d - r_3$, $\forall j \in \{1, \cdots, n_b\}$ where $C^2 \in \mathbb{R}^{n_f \times n_b}$ and $C^4 \in \mathbb{R}^{n_b}$. Meanwhile, Alice computes $C^1_{.i} = X_{.i} + r_1$ and $C^3_i = \sum_{d=1}^{n_f} (r_2 \odot X_{.i})_d + r_3$, $\forall i \in \{1, \cdots, n_a\}$ where $C^1 \in \mathbb{R}^{n_f \times n_a}$ and $C^3 \in \mathbb{R}^{n_a}$. Input-parties send their share of the encoding to the server with the gram matrix of their samples, which is the dot product among their samples. Then, the server computes the dot product between samples of Alice and Bob to complete the missing part of the gram matrix of all samples. To achieve this, the server computes $k_{ij} = \sum_{d=1}^{n_f} (C^1_{.i} \odot C^2_{.j})_d - C^3_i - C^4_j$, $\forall i \in \{1, \cdots, n_a\}$ and $\forall j \in \{1, \cdots, n_b\}$ where k_{ij} is the i-th row j-th column entry of the gram matrix between the samples of the input-parties. Once the server has all components of the gram matrix, it constructs the complete gram matrix K by simply concatenating the parts of it. In our solution, Alice and Bob send to the server (C^1, C^3) and (C^2, C^4) tuples, respectively. These components reveal nothing but only the gram matrix of the samples after decoding. Furthermore, the input-parties shuffle their raw data before the computation to avoid the possibility of private information leakage such as the behavior of the person due to the nature of the visual sequence information. The overall flow is summarized in Figure II.2.

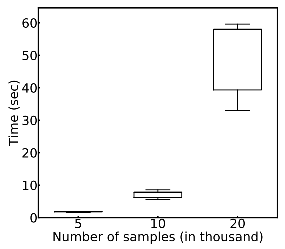
After having the complete gram matrix for all samples that Alice and Bob have, the server uses it as a kernel matrix as if it was computed by the linear kernel function on pooled data. Additionally, it is also possible to compute a kernel matrix as if it was computed by the polynomial or radial basis kernel function (RBF) by utilizing the resulting gram matrix. As an example, the calculation of RBF from the gram matrix is as follows.

$$K(x, y) = \exp\left(- \frac{\|x \cdot x - 2x \cdot y + y \cdot y\|^2}{2\sigma^2} \right),$$

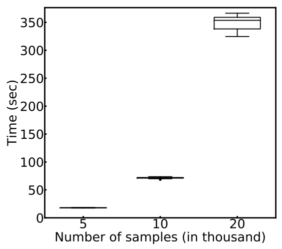
where "·" represents the dot product of vectors, which is possible to obtain from the gram matrix, and σ is the parameter utilized to adjust the similarity level. Once the desired kernel matrix is computed, it is possible to train an SVR model by employing the computed kernel matrix to estimate the gaze. In the process of the computation of the dot product, the amount of data transferred among parties is $(n_f n_a + n_f n_b + n_a + n_b + 2n_f) \times d$ bytes where d is the size of one data unit.



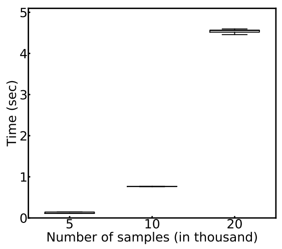
(a) Execution time of Alice.



(b) Execution time of Bob.



(c) Execution time of Server.



(d) Prediction time of Server.

Figure II.3: The execution time of (a) Alice, (b) Bob and (c) the server are given. We also demonstrate (d) the time required for the prediction of the test samples, which are 20% of the total number of samples in each case.

II.4 Security Analysis

A semi-honest adversary who corrupts any of the input-parties cannot learn anything about the private inputs of the other input-party. During the protocol execution, two vectors of random values and a single random value are sent from Alice to Bob. The views of the input-parties consist only of vectors with random values. Using these random values, it is not possible for one party to infer something about the other party's private inputs [86].

Theorem 4. A malicious adversary $\mathscr{A}$ corrupting the function-party learns nothing more than the result of gram matrix. It is computationally infeasible for $\mathscr{A}$ to infer any information about the input-parties' data X and Y as long as Perfect RE multiplication is semantically secure (Definition 4).

Proof. We first show the correctness of our solution. We assume $n_f = 2$ and encode the function $f_d(x, y) = x_1 y_1 + x_2 y_2$ over some finite ring R by the following DARE:

$$\hat{f}_d(x, y; r) = (x_1 + r_{11}, y_1 + r_{12}, x_2 + r_{21}, y_2 + r_{22}, r_{12}x_1 + r_{22}x_2 + r_3,\\ r_{11}y_1 + r_{11}r_{12} + r_{21}y_2 + r_{21}r_{22} - r_3)$$

Given an encoding $(c_1, c_2, c_3, c_4, c_5, c_6)$, $f_d(x, y)$ is recovered by computing $c_1c_2 + c_2c_4 + c_5 + c_6$.

By the concatenation lemma in [24], we can divide c_5 and c_6 into n_f shares by using n_f random values instead of a single r_3 value.

$$\hat{f}_d(x, y; r) = (x_1 + r_{11}, y_1 + r_{12}, r_{12}x_1 + r_{13}, r_{11}y_1 + r_{11}r_{12} - r_{13},\\ x_2 + r_{21}, y_2 + r_{22}, r_{22}x_2 + r_{23}, r_{21}y_2 + r_{21}r_{22} - r_{23})$$

Given an encoding $(c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8)$,

$$\hat{f_m}(x_1, y_1; r) = (c_1, c_2, c_3, c_4)$$
$$\hat{f_m}(x_2, y_2; r) = (c_5, c_6, c_7, c_8)$$

By the concatenation lemma in [24], $\hat{f_d}(x, y; r) = (\hat{f_m}(x_1, y_1; r), \hat{f_m}(x_2, y_2; r))$ perfectly encodes the function $f_d(x, y)$ if Perfect RE multiplication is semantically secure.

After showing the correctness, we analyze the security with the simulation paradigm. In the simulation paradigm, there is a simulator who generates the view of a party in the execution. A party's input and output must be given to the simulator to generate its view. Thus, security is formalized by saying that a party's view can be simulatable given its input and output and the parties learn nothing more than what they can derive from their input and prescribed output.

The function-party $\mathscr{F}$ does not have any input and output. A simulator $\mathscr{S}$ can generate the views of incoming messages received by $\mathscr{F}$. $\mathscr{S}$ creates four vectors $C^{1'}$,$C^{2'}$,$C^{3'}$,$C^{4'}$ with uniformly distributed random values using a pseudorandom number generator G'. Finally, $\mathscr{S}$ outputs $\{C^{1'}, C^{2'}, C^{3'}, C^{4'}\}$.

In the execution of the protocol π, $\mathscr{A}$ receives four messages which are masked with uniformly random values generated using a pseudorandom number generator G. The view of $\mathscr{A}$ includes $\{C^1, C^2, C^3, C^4\}$. The distribution over G is statistically close to the distribution over G'. This implies that

$$\{\mathscr{S}(C^{1'}, C^{2'}, C^{3'}, C^{4'})\} \overset{c}{\equiv} \{view^{\pi}_{\mathscr{A}}(C^1, C^2, C^3, C^4)\}$$

□

II.5 Results

To demonstrate the performance, we conduct experiments on a PC equipped with Intel Core i7-7500U with 2.70 GHz processor and 16 GB memory RAM. We employ varying sizes of eye landmark data, that are $5,000$, $10,000$ and $20,000$ samples of which one-fifth is the test data and we split the data between the input-parties equally. The framework allows us to optimize the parameters of the model in the server without further communicating with the input-parties. Thanks to this, we utilize 5-fold cross-validation to optimize the parameters, which are the similarity adjustment parameter $\gamma \in \{2^{-3}, 2^{-2}, \cdots, 2^4\}$ of the Gaussian RBF kernel, the misclassification penalty parameter $C \in \{2^{-3}, 2^{-2}, \cdots, 2^3\}$, and the tolerance parameter $\epsilon \in \{0.005, 0.01, 0.05, 0.1, 0.5, 1\}$ of SVR. After parameter optimization, we repeat the experiment on varying sizes of eye landmark data with the optimal parameter set 10 times to assess the execution time. To evaluate the gaze estimation results, we employ mean angular error in the same way as in [88]. Table II.1 demonstrates the relationship between the dataset size and the resulting mean angular error. Since no additional noise is introduced during the computation of the kernel matrix, the results from our privacy-preserving framework are the same with the non-private ones. The mean angular errors are lower compared to the state-of-the-art gaze estimation techniques since we use synthetic data and fixed camera position during image rendering.

The amount of time to train and test the models increases as the sample sizes increase since computation requirements get larger. The increment in the dataset size increases the communication

Table II.1: The mean angular errors for varying dataset sizes.

# of samples	Mean angular error
5k	0.21
10k	0.18
20k	0.17

cost among parties. The execution times of all parties for 10 runs with the optimal parameters are shown in Figure II.3. We also demonstrate the amount of time to predict the test samples, which corresponds to one-fifth of the total number of samples to emphasize the real-time working capabilities. In the experiment with $20,000$ samples, for instance, we spend ≈ 4.5 seconds to predict $4,000$ test samples, which corresponds to 1.125 ms per sample. When the current sampling frequencies of eye trackers are taken into consideration, it is possible to deploy and use the framework to estimate gaze if an optimized communication between the parties is established.

II.6 Conclusion

In this work, we utilized a framework based on randomized encoding to estimate human gaze in a privacy-preserving way and in real-time. Our solution can provide improved gaze estimation if input-parties want to use each other's data for different reasons such as to account for genetic structural differences in the eye region. None of the input-parties has the access to the eye landmark data of the others or the result of the computation in the function party, while the function-party cannot infer anything about the data of the input-parties. Temporal information of the visual scanpath, pupillary, or blinks cannot be reconstructed due to the shuffling of the data, and lack of sensory information and direct access to the eye landmarks. Our solution works in real-time, hence it could be deployed along with HMDs for different use-cases and extended to similar eye tracking related problems if similar amount of features is used. To the best of our knowledge, this is the first work based on function-specific privacy models in the eye tracking domain. The number of parties is a limitation of our solution. Thus, as future work we will extend our work to a larger number of parties.

III ESCAPED: Efficient Secure and Private Dot Product Framework for Kernel-based Machine Learning Algorithms with Applications in Healthcare

Ali Burak Ünal **Mete Akgün** **Nico Pfeifer**

Abstract

Training sophisticated machine learning models usually requires many training samples. Especially in healthcare settings these samples can be very expensive, meaning that one institution alone usually does not have enough. Merging privacy-sensitive data from different sources is usually restricted by data security and data protection measures. This can lead to approaches that reduce data quality by putting noise onto the variables (e.g., in ϵ-differential privacy) or omitting certain values (e.g., for k-anonymity). Other measures based on cryptographic methods can lead to very time-consuming computations, which is especially problematic for larger multi-omics data. We address this problem by introducing ESCAPED, which stands for Efficient SeCure And PrivatE Dot product framework. ESCAPED enables the computation of the dot product of vectors from multiple sources on a third-party, which later trains kernel-based machine learning algorithms, while neither sacrificing privacy nor adding noise. We have evaluated our framework on drug resistance prediction for HIV-infected people and multi-omics dimensionality reduction and clustering problems in precision medicine. In terms of execution time, our framework significantly outperforms the best-fitting existing approaches without sacrificing the performance of the algorithm. Even though we only present the benefit for kernel-based algorithms, our framework can open up new research opportunities for further machine learning models that require the dot product of vectors from multiple sources.

III.1 Introduction

In the era of data, the same kind of data is produced by multiple sources. Utilizing this variety of sources is one of the easiest ways to satisfy the hunger of machine learning algorithms for data. Often, one can train a machine learning model on the pooled data from different sources to get high accuracy on a particular prediction task. However, gathering data can compromise the sensitive information of the samples in the data. Ayday et al. [45] showed that genomic data can be used to infer the physical and mental health condition of a patient with the support of information about the patient's lifestyle and environment. Furthermore, Kale et al. [47] introduced a method to keep kinship private in an anonymously released genomic dataset, from which such information could otherwise be inferred. Several studies [46, 89, 90] discussed various privacy issues that occurred in studies using medical data from different aspects.

One class of machine learning methods that usually requires gathering the whole data is kernel-based learning methods. To train such a model privately, one of the architectural models in the literature is the distributed model, where each party in the computation has its own data, and the desired kernel matrix contains the whole data that all parties have. Note that throughout this paper, we refer to a source having data or an entity performing a computation as "*party*". Vaidya et al. [29] proposed an algorithm that uses such a model to compute the gram matrix of the whole data belonging to the parties in the computation and train a support vector machine (SVM) privately afterwards. The disadvantage of the proposed algorithm is that it focuses only on binary vectors because it utilizes private set intersection to compute the dot product. In addition to the distributed model, there is also the outsourced model where the data is outsourced after encryption and then these encrypted data are used to train a kernel-based machine learning method. Liu et al. [52] proposed an approach to use an SVM on the encrypted outsourced data. Due to the nature of encryption, the proposed approach is very time consuming. Zhang et al. [31] introduced a key-switching [32] based secure dot product calculation method. The basic idea is to change the key of the dot product of the vectors, which is originally the combination of the keys utilized to encrypt these vectors, to the key of the server. Ünal et al. [1] demonstrated the inefficiency of this method and proposed a randomized encoding based framework to compute the dot product of the vectors of two parties in a third-party, which later trains an SVM model. However, the framework is not extendable to more than two data sources, since this would compromise the data due to the nature of elementwise multiplication of the vectors, which they use to compute the dot product. Furthermore, for the same reason, their approach has a potential privacy leakage for binary encoded data, even for the case with two data sources. We will show that our approach outperforms their framework in such a scenario. Moreover, the randomized encoding itself [60] is independently applicable to our scenario. The authors claimed that any function expressed by a logarithmic depth arithmetic circuit can be encoded by randomized encoding. In this work, we implemented and applied the randomized encoding based approach and show that it is not as efficient as our framework in terms of the communication cost.

In this paper, we address the privacy problem of data gathering for dot product based algorithms such as kernel-based learning methods. We first implement and apply one of the fastest encodings in the literature, namely the randomized encoding, to our scenario. Due to the inefficiency of the randomized encoding based approach, we come up with a new encoding scheme that enables the secure and private computation of the dot product of vectors. Furthermore, we build a new framework, called efficient secure and private dot product (ESCAPED), which allows multiple data sources, called *input-parties*, to involve in the computation of the dot product. ESCAPED allows a third-party, called *function-party*, to privately obtain the dot product of input-parties' vectors of size larger than 1, while neither gathering the data in plaintext domain nor compromising the privacy of the data. Then, the function-party trains a kernel-based machine learning method. We utilized ESCAPED to predict personalized treatment recommendations for HIV-infected patients in a supervised learning experiment and to perform privacy preserving multi-omics dimensionality reduction and clustering in unsupervised learning experiments. To the best of our knowledge, this is the first study that enables the privacy preserving multi-omics dimensionality reduction and clustering.

III.2 Background

III.2.1 Radial Basis Function Kernel

Among the kernel functions, the radial basis function (RBF) kernel is one of the most effective and widely used kernels [55, 57, 56, 58]. The computation of the RBF kernel for samples $x, y \in \mathbb{R}^n$ can be expressed based on only the dot product of these samples. The formula is as follows:

$$K(x, y) = \exp\left(- \frac{\left\| \langle x, x \rangle - 2\langle x, y \rangle + \langle y, y \rangle \right\|^2}{2\sigma^2} \right) \tag{III.1}$$

where "$\langle \cdot, \cdot \rangle$" represents the dot product of vectors and σ is the parameter that adjusts the similarity level between the samples. Equation III.1 indicates that the gram matrix is enough to compute the RBF kernel. We benefit from such computation to obtain the RBF kernel matrix in ESCAPED.

III.2.2 Randomized Encoding

Randomized encoding (RE) is designed to hide the input value s in the computation of a function $f(s)$ by encoding the function with a randomized function $\hat{f}(s;r)$, where r is a uniformly chosen random value [60, 59]. The decoding of the encoding reveals only the output of the function f but nothing else.

Applebaum [24] introduced the perfect decomposable and affine randomized encoding (DARE) of some operations in their study. For a randomized encoding to be affine and decomposable, all components of the encoding should be affine functions over the set on which the function is defined and each of these components should depend on only a single input value and a varying number of random values. Here, we give only the encodings that we used, which are addition and multiplication-addition operations.

Definition 5 (**Perfect RE for Addition [24]).** Let there be an addition function $t = f(s_1, s_2) = s_1 + s_2$ defined over some finite ring R. The following DARE can perfectly encode such a function:

$$\hat{f}(s_1, s_2; r) = (s_1 + r, s_2 - r)$$

where r is a uniformly chosen random value. The decoding can be done by summing up the components of the encoding, and the simulation of the function can be performed by sampling two random values whose sum is t.

Definition 6 (**Perfect RE for Multiplication-Addition [24]).** Let there be a function $t = f(s_1, s_2, s_3) = s_1 \cdot s_2 + s_3$ defined over a ring R. The following DARE function $\hat{t} = \hat{f}(s_1, s_2, s_3; r_1, r_2, r_3, r_4)$ can perfectly encode the function f:

$$\hat{t} = (s_1 - r_1, r_2 s_1 - r_1 r_2 + r_3, s_2 - r_2, r_1 s_2 + r_4, s_3 - r_3 - r_4)$$

where r_1, r_2, r_3 and r_4 are uniformly chosen random values. Given the encoding $(c_1, c_2, c_3, c_4, c_5)$, the recovery of $f(s_1, s_2, s_3)$ is done by computing $c_1 \cdot c_3 + c_2 + c_4 + c_5$. In order to simulate $\hat{f}$, one can employ the simulator $\mathsf{Sim}(t; c_1, c_2, c_3, c_4) := (c_1, c_2, c_3, c_4, -c_1 c_3 + t - c_2 - c_4)$.

In addition to the given DAREs, the authors claim that any arithmetic circuit with logarithmic depth can be encoded by a perfect DARE [24]. An example of such an arithmetic circuit that computes the dot product of two vectors is given in the Supplement [91]. Taking this into account, we encode the dot product of the vectors by utilizing the aforementioned encodings. Since we only deal with the private computation of the dot product of the vectors, we optimize the generation of the encoding. Let us assume that we have vectors $x, y \in R^D$, where R is a finite ring and $D \in \mathbb{Z}^+$. In the dot product computation, we have D multiplication nodes in the circuit and the results of these multiplication nodes are summed up by using the addition nodes. To generate the encoding of the dot product of vectors x and y, we first find the largest 2's power smaller than D, which we represent here as P, where $2^q = P$ for $q \in \{\mathbb{Z}^+ \cup \{0\}\}$ and $P < D \leq 2 \cdot P$. We separate the summation of the first P of those multiplication nodes from the summation of the remaining $D - P$ multiplication nodes using

Definition 5. We repeat the same procedure for these two parts recursively until we end up with a multiplication node. Once we reach the multiplication node i from an addition node, we utilize the DARE for multiplication-addition given in Definition 6, where $s_1 = x_i$, $s_2 = y_i$ and s_3 represents the resulting value of addition/subtraction of the random values separating summations up to that node. The pseudo code of the randomized encoding generation of the dot product of two vectors of size D and the encoding of the sample arithmetic circuit are given in the Supplement [91].

Randomized encoding has two main applications, namely, secure computing and parallel cryptography. It is commonly used in multi-party computation (MPC) to minimize the round complexity of MPC protocols [92]. Thus, more efficient MPC protocols can be designed using randomized encoding.

III.3 Methods

In this section, we first explain the scenario employed in the paper. Then, we introduce the randomized encoding based approach and our proposed framework ESCAPED. Later, we give the security definition as well as the security analysis of ESCAPED based on the given definition. Finally, we explain the data we used.

III.3.1 Scenario

We consider a scenario where we have multiple input-parties and a function-party, which computes the dot product of vectors of these input-parties and then trains a kernel-based machine learning algorithm. The real life correspondence of such a scenario would be a study in which a researcher wants to employ the same type of data from different patients collected by multiple hospitals, like cancer subtype discovery. In this scenario, one would like to group cancer patients according to similarities with respect to their omics data. For a new patient, the subtype could give first hints about how severe the cancer is and how well the prognosis is with regard to potential treatments and life expectancy. Due to patient privacy, such data cannot be shared without a permission process, which can significantly slow down the study. However, a framework, like ESCAPED, ensures the protection of the privacy of patients' data, hence enabling the researcher to speed up permission processes and enables studies that would otherwise not be approved due to privacy concerns. While describing the approaches, even though both the randomized encoding based approach and ESCAPED can have multiple input-parties, for simplicity, we use a scenario with three input-parties, namely, Alice, Bob and Charlie with ids 1, 2 and 3, respectively, and a function-party.

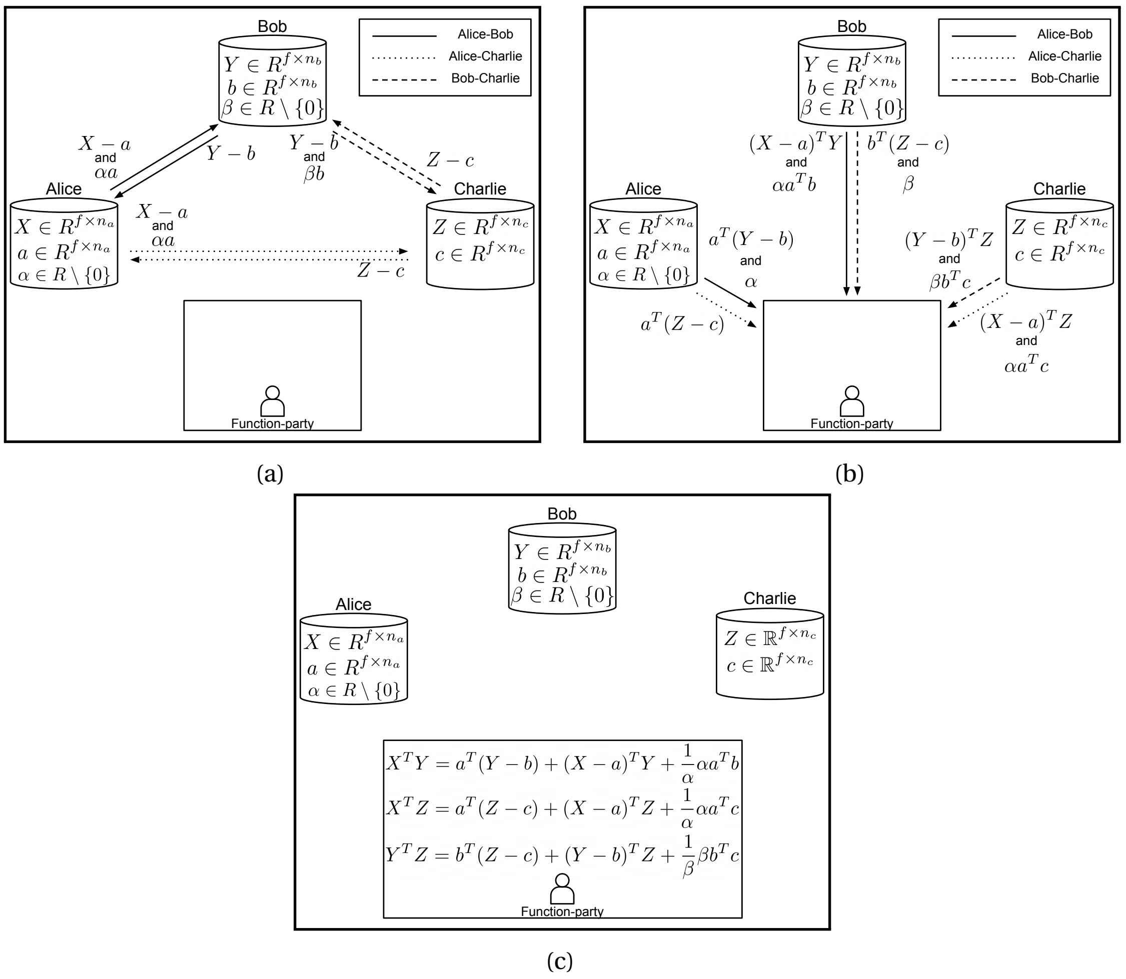
III.3.2 Randomized Encoding Based Approach

To address the aforementioned problem, we first implemented a randomized encoding based approach and applied it to our scenario. In this scenario, each of the input-parties has their own data $X \in R^{f \times n_a}$, $Y \in R^{f \times n_b}$ and $Z \in R^{f \times n_c}$, respectively, where f represents the number of features, n_x represents the number of samples in the corresponding input-party and R is a finite ring. Each pair of input-parties needs to communicate separately, i.e., there is communication between Alice and Bob, Alice and Charlie, and Bob and Charlie. For simplicity, we explain only the communication between Alice and Bob. To compute $X^T Y$, they first exchange the size of their own data. Afterwards, Alice generates the scheme of the encoding of the dot product by utilizing the randomized encoding generation algorithm given in the Supplement [91]. Using the resulting encoding scheme, she creates a new set of random values for each possible pair of samples, consisting of one sample from Alice and one sample from Bob. This is quite important in order to protect the relative difference of the features of the input-parties' samples from the function-party. For instance, using the component $s_1 - r_1$ in the encoding, the function-party could learn the relative differences of the input values in the case that the same random value r_1 is utilized for more than one pair of samples. Once Alice created all random values, she sends Bob the part of these random values that he will use to encode his own data. Afterwards, both Alice and Bob encode their data by employing the corresponding random values and send the resulting components to the function-party along with the gram matrix of their own samples. To compute the dot product of samples of Alice and Bob, the function-party combines these components according to the decoding described in Definitions 5 and 6. Such communication is done between all possible pairs of the input-parties, which means that if we have M input-parties, there will be $\binom{M}{2}$ communications in total (more detailed communication cost analysis in Table III.1). Once the function-party has all partial gram matrices, it constructs the gram matrix by vertically concatenating the horizontally concatenated partial gram matrices $[X^T X, X^T Y, X^T Z]$, $[Y^T X, Y^T Y, Y^T Z]$ and $[Z^T X, Z^T Y, Z^T Z]$. Then it can compute the desired kernel matrix, which can be computed via the gram matrix, and train a kernel-based machine learning method. The overview of the dot product computation procedure via the randomized encoding based approach is given in the Supplement [91]. Note that in the supervised scenario the input-parties share the labels of the samples with the function-party in plaintext domain since they do not reveal any extra and sensitive information. However, this could easily be extended if more sensitive labels are supposed to be used in the learning process.

III.3.3 ESCAPED

Due to the high communication cost of the randomized encoding based approach to securely compute the dot product of vectors from multiple input-parties in the function-party, we propose a new, efficient and secure framework, called ESCAPED, which is based on a new encoding scheme for the dot product computation. In the computation, the input-parties do not learn anything about the data of the other input-parties or the result of any dot product computed by the function-party. Similarly, the function-party learns only the dot product of the data from the input-parties, but nothing else.

For simplicity, we explain only the computation of the dot product of the data from Alice and Bob in ESCAPED. Figure III.1 depicts the overview of ESCAPED. First, Alice and Bob create matrices of random values $a \in R^{f \times n_a}$ and $b \in R^{f \times n_b}$, respectively, where R is a finite ring. Along with these random valued matrices, Alice also creates a random value $\alpha \in R \setminus \{0\}$. Afterwards, Alice computes $X - a$ and αa, and shares them with Bob. In the meantime, Bob computes $Y - b$ and sends it to Alice. Once Alice receives the masked data of Bob, she computes $A_1 = a^T (Y - b)$. Meanwhile, Bob computes $B_1 = (X - a)^T Y$ and $B_2 = \alpha a^T b$. Then, Alice sends A_1 and α, and Bob sends B_1 and B_2 along with the gram matrix of their own samples, which are $X^T X$ and $Y^T Y$, respectively, to the function-party. At this point, the function-party computes $A_1 + B_1 + \frac{1}{\alpha} B_2$ to obtain the dot product of the data of Alice and Bob, which is $X^T Y$. Such communication is done similarly among all pairs of input-parties. In these communications, the input-party with a smaller id becomes *"Alice"* and the other becomes *"Bob"*. In the end, the function-party has the gram matrix of all samples. Afterwards,



Bob
$Y \in R^{f \times n_b}$
$b \in R^{f \times n_b}$
$\beta \in R \setminus \{0\}$
Alice
$X \in R^{f \times n_a}$
$a \in R^{f \times n_a}$
$\alpha \in R \setminus \{0\}$
Charlie
$Z \in R^{f \times n_c}$
$c \in R^{f \times n_c}$
Alice-Bob
Alice-Charlie
Bob-Charlie
$X - a$ and αa
$Y - b$
$Y - b$ and βb
$Z - c$
$X - a$ and αa
$Z - c$
Function-party
(a)
$(X - a)^T Y$ and $\alpha a^T b$
$b^T (Z - c)$ and β
$a^T (Y - b)$ and α
$(Y - b)^T Z$ and $\beta b^T c$
$a^T (Z - c)$
$(X - a)^T Z$ and $\alpha a^T c$
(b)
$X^T Y = a^T (Y - b) + (X - a)^T Y + \frac{1}{\alpha} \alpha a^T b$
$X^T Z = a^T (Z - c) + (X - a)^T Z + \frac{1}{\alpha} \alpha a^T c$
$Y^T Z = b^T (Z - c) + (Y - b)^T Z + \frac{1}{\beta} \beta b^T c$
(c)

Figure III.1: The overview of ESCAPED in our scenario. Each dash type corresponds to a specific part of the gram matrix computed by a pair of input-parties. **(a)** First, the input-parties exchange their masked input data (e.g. $X - a$) and masked masks (e.g. αa), if applicable. **(b)** Then, they compute the components of all dot products they are responsible for (e.g. $a^T(Y - b)$) and send them to the function-party along with the mask of the mask (e.g. α), if applicable. **(c)** The function-party computes the dot product based on the corresponding components of the input-parties.

the function-party can compute the desired kernel matrix like the RBF kernel matrix, which can be calculated by using Equation III.1, and train a kernel-based machine learning method using the computed kernel matrix to obtain a prediction model. Note that the input-parties share the labels with the function-party in the plaintext domain for of the same reason we mentioned earlier.

Table III.1 summarizes the features and the communication cost analysis of ESCAPED, the randomized encoding based approach and the approach proposed by Ünal et al. [1].

III.3.4 Security Definition

In our proof, we utilize two different adversarial models, which are the *semi-honest* adversary model, or *honest-but-curious*, and the *malicious* adversary model. A *semi-honest* adversary is a computationally bounded adversary that follows the protocol strictly but also tries to infer any valuable information from the messages seen during the protocol execution. On the other hand, in the *malicious* adversary model, a *malicious* adversary can arbitrarily deviate from the protocol specification. Although the semi-honest model has more restrictive assumptions than the malicious model, it makes the development of highly efficient privacy preserving protocols relatively easy.

Let there be M input-parties $(\mathscr{I}_1, ..., \mathscr{I}_M)$ and a function-party $\mathscr{F}$ in the proposed system. We assume that an adversary is either a semi-honest adversary corrupting a subset of input-parties or a malicious adversary corrupting the function-party. We restrict the collusion between the function-party and the input-parties so as not to allow the corruption of the function-party and at least one input-party at the same time. Otherwise, an adversary $\mathscr{A}$ who corrupts the function-party and at least one input-party obtains the inputs of all other input-parties. Even though we allow the collusion among input-parties, one might think that it is not so realistic because involved entities, such as medical institutions, lose their reputations if they misbehave in this setting.

We use the simulation paradigm [93] in our security proofs. In the simulation paradigm, the security is proven by showing that the simulator can simulate the input and the output of a party, given the actual input and output, such that the simulated input and output cannot be distinguished from the actual ones by an observer. Such an indistinguishability indicates that the parties cannot learn more than what can be learned from their inputs and outputs.

The function-party constructs the final output, i.e. the gram matrix, by using the partial outputs each of which is computed by a pair of input-parties. This enables us to consider these computations as a separate two-party computation. The following notations are used in the security definition:

- Let $f = (f_1, f_2)$ be a probabilistic polynomial-time functionality, where f_p is the input provided by the p-th party to f and let π be a two-party protocol for computing f.
- The view of the i-th party ($i \in 1,2$) during an execution of π over (x, y) is denoted by $v_i^\pi(x, y)$ and equals $(w, r^i, m_1^i, ..., m_t^i)$, where $w \in \{x, y\}$, r^i equals the contents of the i-th party's internal random tape and m_j^i represents the j-th message it received.
- The output of the i-th party during an execution of π over (x, y) is denoted by $o_i^\pi(x, y)$ and can be computed from its own view of the execution. We denote the joint output of both parties by $o^\pi(x, y) = (o_1^\pi(x, y), o_2^\pi(x, y))$.

Definition 7. Let $f = (f_1, f_2)$ be a functionality. We say that a protocol π is secure against semi-honest adversaries if there exist probabilistic polynomial time (PPT) simulators S_1 and S_2 such that:

$$(S_1(x, f_1(x, y)), f(x, y)) \overset{c}{\equiv} (v_1^\pi(x, y), o_1^\pi(x, y))$$

$$(S_2(y, f_2(x, y)), f(x, y)) \overset{c}{\equiv} (v_2^\pi(x, y), o_2^\pi(x, y))$$

where $\overset{c}{\equiv}$ denotes the computational indistinguishability. More details can be found in [94].

III.3.5 Security Analysis

Theorem 5. ESCAPED is secure against a semi-honest adversary $\mathscr{A}$ that corrupts any subset of input-parties.

Proof. The proof is provided in the Supplement [91]. □

Theorem 6. Assume that the function-party is malicious and does not collude with any input-parties. Then, ESCAPED is secure against the malicious function-party $\mathscr{A}$ such that $\mathscr{A}$ cannot infer the data of input-parties from neither the components sent by the input-parties nor the resulting gram matrix.

Proof. The proof is provided in the Supplement [91]. □

III.3.6 Data

In this section, we briefly explain the datasets we employed in our supervised and unsupervised learning experiments, respectively.

***HIV V3 Loop Sequence Dataset*:** To predict the personalized treatmet of HIV-infected patients in the supervised learning experiments, we retrieved the HIV V3 loop dataset from Ünal et al. [1]. It consists of the protein sequence of the viruses as well as their coreceptor usage information. Due to the availability of drugs blocking the human CCR5 coreceptor, which is exclusively used by the most common variant of HIV to enter the cell, identifying the coreceptor usage is crucial for determining whether or not to use these drugs [61]. The dataset consists of 642 samples for the class "CCR5 only" and 124 samples for the class "OTHER". The sequence data exists as a one-hot encoded data matrix with 766 rows and 924 columns.

***Head and Neck Squamous Cell Carcinoma Dataset*:** We aim to perform the privacy preserving multi-omics dimensionality reduction and clustering on the TCGA data for head and neck squamous cell carcinoma (HNSC) [95] to stratify patients into clinically meaningful subgroups. Therefore, we replicate a recent state-of-the-art study [40] in a privacy-preserving setting, obtaining the data from the authors. The data consists of 465 patients with their gene expression (IlluminaHiSeq), DNA methylation (Methylation450k), copy number variation (gistic2), and miRNA expression (IlluminaHiSeq) data. They have 19433, 57159, 23817 and 581 features, respectively. We also obtained the survival times of the patients.

III.4 Results

In order to simulate multiple input-parties, we created a process for each input-party and shared the data among them equally. We also created an additional process to simulate the function-party. All processes communicate with each other over TCP sockets and we assume that the communication is secure. We conducted the experiments on a server with has 512 GB memory, an Intel Xeon E5-2650 processor and a 64-bit operating system. We utilized Python to implement ESCAPED and the randomized encoding based approach.

Approach	Number of IPs		Communication cost		
	Two IPs	Three or more IPs	Among IPS	Between IPs and FP	Total
UAP	Yes	No	$3R^{f\times n^2}$ *	$4R^{f\times n^2}$ *	$7R^{f\times n^2}$ *
RE	Yes	Yes	$4\binom{M}{2}R^{f\times n^2}$	$5\binom{M}{2}R^{f\times n^2}$	$9\binom{M}{2}R^{f\times n^2}$
ESCAPED	Yes	Yes	$3\binom{M}{2}R^{f\times n}$	$3\binom{M}{2}R^{n^2}$	$3\binom{M}{2}(R^{f\times n}+R^{n^2})$

Table III.1: The summary of the comparison of the methods utilized in this study from different aspects. The first part of the table presents the ability to handle a varying number of input-parties (IP) in the framework proposed by Ünal et al. [1] (UAP), the randomized encoding based approach (RE) and ESCAPED. Moreover, n being the number of samples in each IP, M being the number of IPs and f being the number of features of samples, where $n, M, f \in \mathbb{Z}^+$ and $M \geq 2$, the second part of the table presents the communication cost analysis of RE and ESCAPED in terms of the communication cost among IPs, between IP and the function-party (FP) and the total communication cost. The communication cost analysis of UAP, however, is given without any dependency on M since it can only handle two input-parties scenario. Note that we omit the communication cost of sending the gram matrix of the samples belonging to the same IP since it is fixed for all approaches.

III.4.1 Classification of HIV Coreceptor Usage

In these supervised learning experiments, we used an SVM with an RBF kernel matrix. We optimized the parameters of the SVM, which are the misclassification penalty $C \in \{2^{-5}, 2^{-4}, \cdots, 2^{10}\}$ and the weight $w_1 \in \{2^0, 2^1, \cdots, 2^5\}$ of the minority class, and the similarity adjustment parameter $\sigma \in \{2^{-5}, 2^{-4}, \cdots, 2^{10}\}$ of the RBF kernel via 5-fold cross-validation and F1-score. Note that we tuned the parameters outside of the approaches and used them in the experiments directly. However, one can employ both ESCAPED and the randomized encoding based approach for tuning. To have a fair evaluation, we repeated the optimization step 10 times with different random folds and conducted separate experiments by using each optimal parameter set. We evaluated the experiments via F1-score and area under receiver operating characteristic curve (AUROC).

We utilized our proposed framework, ESCAPED, to compute the dot product of samples of three different input-parties on a function-party. Once the function-party has the gram matrix, it computes the RBF kernel matrix based on the optimal σ using Equation III.1. We separated 20% of the data of each input-party for testing. The function-party trains an SVM model on the rest of the data by employing the optimal parameters w_1 and C. Then, we tested the model on the test data. Finally, we evaluated the prediction of our model via F1-score and AUROC. We repeated this experiment for each optimal parameter set and obtained 0.843 (±0.013) AUROC and 0.615 (±0.016) F1-score on average. To demonstrate the scalability of ESCAPED in terms of the total dataset size, we conducted experiments in which we used a quarter, a half and the full dataset. The execution time of ESCAPED increases almost quadratically with respect to the size of the dataset. Figure III.2a shows the trend of

*The communication cost analysis is given after an update on UAP to protect the privacy of relative differences between features of samples. Without any update, the communication costs would become $3R^f$, $4R^{f\times n}$ and $3R^f + 4R^{f\times n}$, respectively.

the increase in the execution time in parallel to the increment in the dataset size. Furthermore, we analyzed the performance of the framework for varying number of input-parties each of which has the same number of samples. Figure III.2b displays the effect of the number of input-parties involved in the computation on the execution time of various parts. The total execution time and the total communication time between input-parties and the function-party (black and red, respectively) are almost linear. The total communication among input-parties (orange), however, displays a slightly different pattern. Since there is an idle party in each turn of the communication among input-parties when there is an odd number of input-parties, the execution time for the cases with an even number of input-parties is almost the same as the case where we have one less input-party.

We also applied the randomized encoding based approach to the same scenario. Similar to the experiments with ESCAPED, we repeated the whole experiment for each optimal parameter set. Since we obtained exactly the same F1-score and AUROC with ESCAPED for the same set of parameters, we demonstrate only the execution time of the randomized encoding based approach for the varying size of the dataset in Figure III.2c. When we compared the execution time of the randomized encoding based approach to ESCAPED for full dataset experiments, the randomized encoding based approach took 1.3×10^4 $(\pm 1.4 \times 10^3)$ *sec* whereas ESCAPED took only 1.19×10^1 $(\pm 3.5 \times 10^{-2})$ *sec*. Since the randomized encoding based approach is quite inefficient compared to ESCAPED, we did not evaluate it in terms of the number of input-parties. Based on the results and the cost analysis shown in Table III.1, it is fair to claim that ESCAPED is more efficient than the randomized encoding approach and other MPC protocols in the literature.

Even though Ünal et al. [1] cannot handle three or more input-parties, we compared ESCAPED to this framework in case of a scenario with two input-parties. Since we obtained the same results for both methods, we only give the execution time comparison of them. Figure III.2d shows that ESCAPED outperforms their framework. Based on this observation, we can state that ESCAPED is more efficient and comprehensive, especially considering its applicability to more than two input-parties.

III.4.2 Clustering of HNSC Cancer Patients

To demonstrate the applicability of ESCAPED on unsupervised learning problems on multi-view data, we employed it to determine biologically meaningful subgroups of cancer patients. Speicher and Pfeifer [8] studied such a problem and suggested a regularized multiple kernel learning algorithm with dimensionality reduction (rMKL-DR). The method was recently evaluated as the best method in a large benchmark study that compared many different methods [41]. Later, Röder et al. [40] published the online version of the method called web-rMKL. In that study, one of their use cases is the identification of subgroups of HNSC. To stratify patients into biologically meaningful subgroups, they employed four different data types: gene expression, DNA methylation, miRNA expression and copy number variation. They computed one RBF kernel matrix, whose γ is chosen based on a rule of thumb, for each data type and input these kernel matrices to the web-rMKL to obtain the subgroups of patients. They pruned patients whose survival is longer than 5 years. Then they evaluated the results by survival analysis and obtained a $p = 0.0006$ in log-rank test. To show the applicability of ESCAPED, we replicated their study in a privacy preserving way. We utilized the same dataset and split it equally into three input-parties. We employed ESCAPED to compute the kernel matrix for each data type, based on the data belonging to different input-parties. It took 129.17 (± 3.81) *sec* to

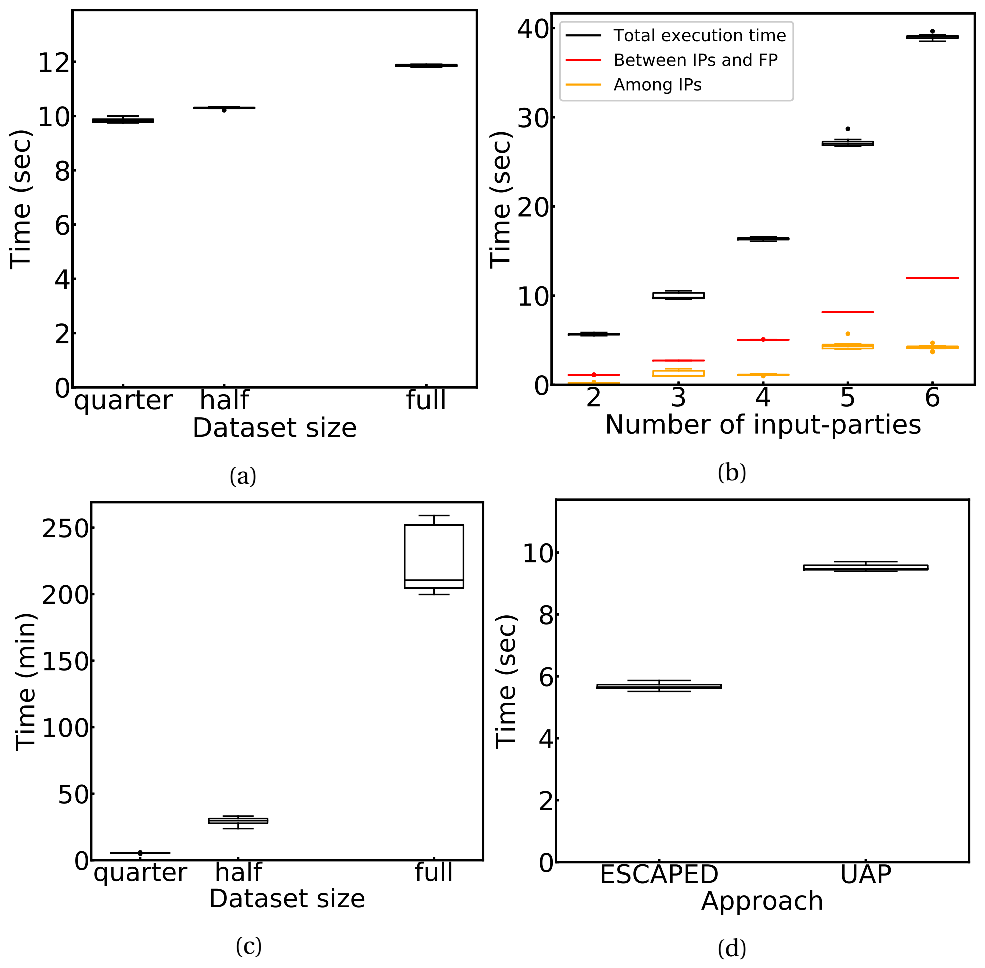


Figure III.2: **(a)** The execution time of ESCAPED is shown for varying sizes of the dataset. **(b)** The analysis of ESCAPED for varying number of input-parties in terms of the total execution time, the communication time between the input-parties (IP) and the function-party (FP), and the communication time among IPs is shown. **(c)** The execution time of the randomized encoding based approach is depicted for different sizes of the dataset. The execution time increases quadratically in parallel to the increment in the dataset size. **(d)** The execution time comparison of ESCAPED and the UAP [1] for two input-parties case is given.

compute the required kernel matrices. We then input the resulting kernel matrices to web-rMKL with the same parameter choices to cluster patients. We applied the same filters and evaluated the results by survival analysis as they did. In the end, we obtained the same p-value, indicating that ESCAPED is capable of performing privacy preserving multi-omics dimensionality reduction and clustering. We were unable to conduct these experiments via the randomized encoding based approach due to the excessive memory usage stemming from the inefficiency of the randomized encoding on high dimensional data.

III.5 Conclusion

The tension between the unavoidable demand of machine learning algorithms for data and the importance of the privacy of the sensitive information in data urges researchers to come up with efficient and privacy preserving machine learning algorithms. To address this necessity, we introduced ESCAPED to enable the secure and private computation of the dot product in our scenario. In ESCAPED, we preserve the privacy of the data in the computation while neither sacrificing the performance of the model nor adding noise. We demonstrated the efficiency and applicability of ESCAPED on the personalized treatment prediction system of HIV-infected patients and the privacy preserving multi-omics dimensionality reduction and clustering of HNSC patients into biologically meaningful subgroups. Also, we implemented and applied the randomized encoding based approach to solve these problems securely. In the supervised learning problem, both approaches yielded the same result in terms of F1-score and AUROC, but ESCAPED outperformed the randomized encoding based approach in terms of execution time. In the unsupervised learning case, we replicated the state-of-the-art experiments conducted by Röder et al. [40] in a privacy preserving way without sacrificing performance. This indicates that ESCAPED enables performing privacy preserving multi-omics dimensionality reduction and clustering whereas it was not possible to compute the required kernel matrices with the randomized encoding based approach, which is one of the fastest competitors, due to the excessive memory usage. Even though we applied ESCAPED to two machine learning methods, it is applicable to any method requiring the dot product of the vectors from multiple sources on a third-party, showing the promise of efficiently making other learning algorithms privacy preserving as well. As a future work, other commonly used operations in machine learning algorithms could be included in the framework to extend the scope of the framework. Furthermore, the interpretability of the resulting model could be improved to allow more sophisticated analyses via the model.

Acknowledgements

This study is supported by the DFG Cluster of Excellence “Machine Learning – New Perspectives for Science”, EXC 2064/1, project number 390727645. Furthermore, NP and MA acknowledge funding from the German Federal Ministry of Education and Research (BMBF) within the ’Medical Informatics Initiative’ (DIFUTURE, reference number 01ZZ1804D).

Ethics Statement

Thanks to the promising results of ESCAPED, our study could open up new collaboration opportunities among hospitals, universities, institutes, data centers and many other entities with faster permission processes by providing secure and private computation of dot product enabling, not only the kernel-based learning algorithms but also other methods requiring the dot product. This would help to speed up healthcare research that helps humanity and the world in general. We could not think of a negative ethical impact of our work.

III.6 Supplement

III.6.1 Randomized Encoding Generation Algorithm

The pseudo code of the randomized encoding generation of the dot product of two vectors of size D is given in Algorithm III.1.

```
1  Algorithm REGen()
       input  : $eY$: the list of the indices of the random values for Y
                $eO$: the list of the indices of the random values for the offline part
                $eOS$: the list of the sign of the random values of the offline part
                $R$: the starting value of the random values that will be generated
       output : $R$: the latest value of the generated random values
2      $D \leftarrow$ length of $eX$
3      if $D = 1$ then
4          $eX[0] \leftarrow [0, R{+}1, R, R, R{+}1, R{+}2]$ ▷ In the generated random values afterwards, the first
             one, indicated by 0 here, is always 1
5          $eY[0] \leftarrow [0, R, R{+}1, R{+}3]$
6          $eO[0] \leftarrow [eO[0], R{+}2, R{+}3]$
7          $eOS[0] \leftarrow [eOS[0], -1, -1]$
8          $R \leftarrow R + 4$
9          return $R$
10     $q \leftarrow \mathrm{argmax}_q\,(2^q < D)$                    ▷ $q \in \{\mathbb{Z}^+ \cup \{0\}\}$
11     $P \leftarrow 2^q$
12     $eO[0] \leftarrow [eO[0], R]$
13     $eOS[0] \leftarrow [eOS[0], +1]$                ▷ +1 represents positive sign
14     $R \leftarrow R + 1$
15     $R \leftarrow REGen(eX[0:P], eY[0:P], eO[0:P], eOS[0:P], R)$
16     $R \leftarrow REGen(eX[P:D], eY[P:D], eO[P:D], eOS[P:D], R)$
17     return $R$
```

Algorithm III.1: Randomized Encoding Generation for Dot Product

III.6.2 Sample Encoding

A sample arithmetic circuit computing the dot product of two vectors is given in Figure III.3, where we use two vectors of size 7. The encoding of the corresponding circuit is given in Equation III.2.

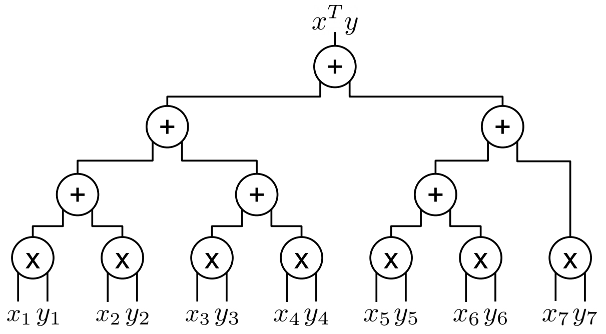


Figure III.3: An arithmetic circuit to compute the dot product of the vectors x and y where $x, y \in \mathbb{R}^7$.

$$\begin{aligned}
\hat{f}(x,y,R) = \Bigg(& x_1 \begin{bmatrix} 1 \\ r_7 \end{bmatrix} + \begin{bmatrix} -r_6 \\ -r_6 r_7 + r_8 \end{bmatrix}, y_1 \begin{bmatrix} 1 \\ r_6 \end{bmatrix} + \begin{bmatrix} -r_7 \\ r_9 \end{bmatrix}, r_0 + r_1 + r_3 - r_8 - r_9, \\
& x_2 \begin{bmatrix} 1 \\ r_{11} \end{bmatrix} + \begin{bmatrix} -r_{10} \\ -r_{10} r_{11} + r_{12} \end{bmatrix}, y_2 \begin{bmatrix} 1 \\ r_{10} \end{bmatrix} + \begin{bmatrix} -r_{11} \\ r_{13} \end{bmatrix}, -r_3 - r_{12} + r_{13}, \\
& x_3 \begin{bmatrix} 1 \\ r_{15} \end{bmatrix} + \begin{bmatrix} -r_{14} \\ -r_{14} r_{15} + r_{16} \end{bmatrix}, y_3 \begin{bmatrix} 1 \\ r_{14} \end{bmatrix} + \begin{bmatrix} -r_{15} \\ r_{17} \end{bmatrix}, -r_1 + r_4 - r_{16} + r_{17}, \\
& x_4 \begin{bmatrix} 1 \\ r_{19} \end{bmatrix} + \begin{bmatrix} -r_{18} \\ -r_{18} r_{19} + r_{20} \end{bmatrix}, y_4 \begin{bmatrix} 1 \\ r_{18} \end{bmatrix} + \begin{bmatrix} -r_{19} \\ r_{21} \end{bmatrix}, -r_4 - r_{20} - r_{21}, \\
& x_5 \begin{bmatrix} 1 \\ r_{23} \end{bmatrix} + \begin{bmatrix} -r_{22} \\ -r_{22} r_{23} + r_{24} \end{bmatrix}, y_5 \begin{bmatrix} 1 \\ r_{22} \end{bmatrix} + \begin{bmatrix} -r_{23} \\ r_{25} \end{bmatrix}, -r_0 + r_2 + r_5 - r_{24} - r_{25}, \\
& x_6 \begin{bmatrix} 1 \\ r_{27} \end{bmatrix} + \begin{bmatrix} -r_{26} \\ -r_{26} r_{27} + r_{28} \end{bmatrix}, y_6 \begin{bmatrix} 1 \\ r_{26} \end{bmatrix} + \begin{bmatrix} -r_{27} \\ r_{29} \end{bmatrix}, -r_5 - r_{28} - r_{29}, \\
& x_7 \begin{bmatrix} 1 \\ r_{31} \end{bmatrix} + \begin{bmatrix} -r_{30} \\ -r_{30} r_{31} + r_{32} \end{bmatrix}, y_7 \begin{bmatrix} 1 \\ r_{30} \end{bmatrix} + \begin{bmatrix} -r_{31} \\ r_{33} \end{bmatrix}, -r_2 - r_{32} - r_{33} \Bigg)
\end{aligned} \tag{III.2}$$

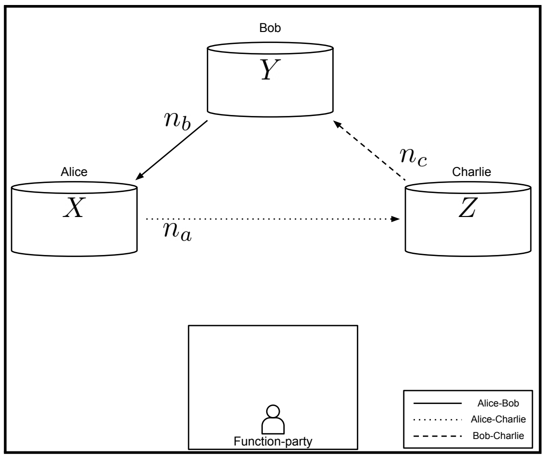
Let $\hat{f}(x,y,R)$ be $\left(\begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \begin{bmatrix} c_3 \\ c_4 \end{bmatrix}, c_5, \ldots, \begin{bmatrix} c_{31} \\ c_{32} \end{bmatrix}, \begin{bmatrix} c_{33} \\ c_{34} \end{bmatrix}, c_{35} \right)$, decoding of $\hat{f}$ is done by computing the following:

$$f(x,y) = x^T y = \sum_{i \in S} c_i * c_{i+2} + c_{i+1} + c_{i+3} + c_{i+4}$$

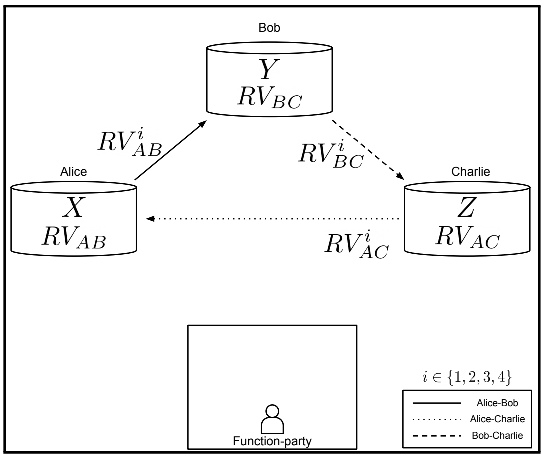
where $S = \{1, 6, 11, 16, 21, 26, 31\}$ is the set of indices indicating the beginning of each encoded multiplication $x_i \cdot y_i$ for $i \in \{1, 2, \ldots, 7\}$.

III.6.3 The Scheme of Randomized Encoding Based Approach

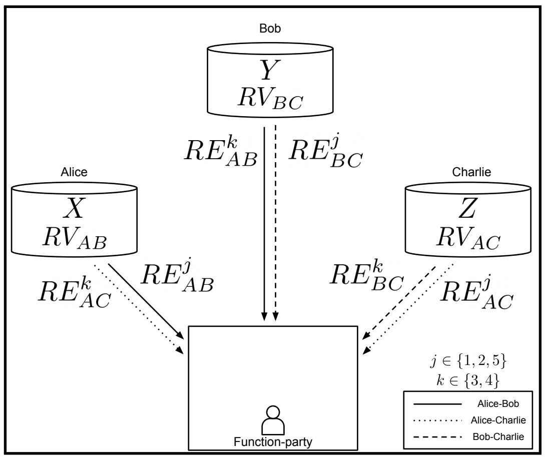
The scheme of the randomized encoding based approach to compute the gram matrix is depicted in Figure III.4.



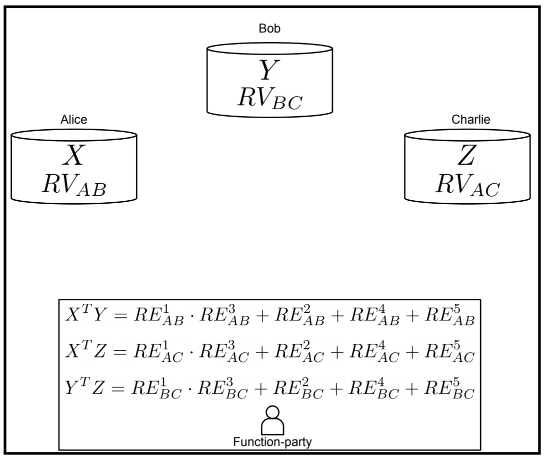
(a)



(b)



(c)


Bob
Y RV_{BC}
Alice
X RV_{AB}
Charlie
Z RV_{AC}
$X^TY = RE^1_{AB} \cdot RE^3_{AB} + RE^2_{AB} + RE^4_{AB} + RE^5_{AB}$
$X^TZ = RE^1_{AC} \cdot RE^3_{AC} + RE^2_{AC} + RE^4_{AC} + RE^5_{AC}$
$Y^TZ = RE^1_{BC} \cdot RE^3_{BC} + RE^2_{BC} + RE^4_{BC} + RE^5_{BC}$
Function-party

(d)

$X \in \mathbb{R}^{f \times n_a}$	$RV_{AB} \in \mathbb{R}^{5f \times (n_a \cdot n_b)}$	$RV^i_{AB} \in \mathbb{R}^{f \times (n_a \cdot n_b)}$	$RE^j_{AB} \in \mathbb{R}^{f \times (n_a \cdot n_b)}$	$RE^k_{BC} \in \mathbb{R}^{f \times (n_b \cdot n_c)}$
$Y \in \mathbb{R}^{f \times n_b}$	$RV_{BC} \in \mathbb{R}^{5f \times (n_b \cdot n_c)}$	$RV^i_{BC} \in \mathbb{R}^{f \times (n_b \cdot n_c)}$	$RE^k_{AB} \in \mathbb{R}^{f \times (n_a \cdot n_b)}$	$RE^j_{AC} \in \mathbb{R}^{f \times (n_a \cdot n_c)}$
$Z \in \mathbb{R}^{f \times n_c}$	$RV_{AC} \in \mathbb{R}^{5f \times (n_a \cdot n_c)}$	$RV^i_{AC} \in \mathbb{R}^{f \times (n_a \cdot n_c)}$	$RE^j_{BC} \in \mathbb{R}^{f \times (n_b \cdot n_c)}$	$RE^k_{AC} \in \mathbb{R}^{f \times (n_a \cdot n_c)}$

(e)

Figure III.4: The overview of the randomized encoding based approach to compute the dot product of samples from multiple input-parties is depicted. Each dash type corresponds to a specific part of the gram matrix computed by a pair of input-parties. **(a)** At first, the input-parties exchange their number of samples. **(b)** Then, they generate all the random values and send the required ones to the corresponding input-party. **(c)** Afterwards, they compute their components in the encoding and share them with the function-party. **(d)** Finally, the function-party computes the dot product of the samples. **(e)** The dimension of the matrices are shown separately for better readability.

III.6.4 Security Proof

Theorem 7. Efficient secure and private dot product framework (ESCAPED) is secure against a semi-honest adversary $\mathscr{A}$ that corrupts any subset of input-parties.

Proof. According to the definition of the semi-honest adversary, $\mathscr{A}$ cannot deviate from the protocol description. Thus, $\mathscr{A}$ has to use the real inputs of the corrupted input-parties. Based on this information, it is easy to prove the correctness of ESCAPED. For simplicity, we present our proof using a scenario with three input-parties ($\mathscr{I}_1, \mathscr{I}_2, \mathscr{I}_3$). The private data of $\mathscr{I}_1, \mathscr{I}_2$, and $\mathscr{I}_3$ are X, Y, and Z, respectively, and the function-party $\mathscr{F}$ wants to learn X^TY, X^TZ, and Y^TZ. Equation III.3 shows the correctness of ESCAPED.

$$\begin{aligned}
X^TY &= a^T(Y-b) + (X-a)^TY + \frac{1}{\alpha}\alpha a^Tb \\
&= a^TY - a^Tb + X^TY - a^TY + a^Tb \\
&= X^TY \\
X^TZ &= a^T(Z-c) + (X-a)^TZ + \frac{1}{\alpha}\alpha a^Tc \\
&= a^TZ - a^Tc + X^TZ - a^TZ + a^Tc \\
&= X^TZ \\
Y^TZ &= b^T(Z-c) + (Y-b)^TZ + \frac{1}{\beta}\beta b^Tc \\
&= b^TZ - b^Tc + Y^TZ - b^TZ + b^Tc \\
&= Y^TZ
\end{aligned} \tag{III.3}$$

After showing the correctness, we now prove the security of ESCAPED. $\mathscr{I}_1, \mathscr{I}_2$, and $\mathscr{I}_3$ learn ($Y-b$, $Z-c$), ($X-a$, αa, $Z-c$), and ($X-a$, αa, $Y-b$, βb), respectively. These values are generated by using random values α and β, and matrices of uniformly chosen random values, a, b and c. Thus, they can be perfectly simulated by matrices of uniformly random values, which indicates the security of ESCAPED. More clearly, an input-party gets the randomly masked input data of other input-parties as well as their randomly masked masks. For instance, $\mathscr{I}_2$ receives $(X-a), (Z-c)$ and αa. Since both the mask of the mask, α, and the mask of the data, a and b, are uniformly random, it is impossible for $\mathscr{I}_2$ to learn the input data of other input-parties, which are X and Z in this case. □

Theorem 8. Assume that the function-party is malicious and does not collude with any input-parties. Then, ESCAPED is secure against the malicious function-party $\mathscr{A}$ such that $\mathscr{A}$ cannot infer the data of input-parties from neither the components sent by the input-parties nor the resulting gram matrix.

Proof. Since the function-party does not have any input, it cannot change the output of any input-parties. This guarantees and proves the correctness of ESCAPED against the malicious function-party.

The security of the framework from the perspective of the function-party leans on two facts. The first one is that the number of features of the input data is unknown for the function-party. $\mathscr{A}$ cannot be sure which vector space the samples are coming from. This makes unique prediction impossible. The second one is that $\mathscr{A}$ gets the gram matrix of input data among input-parties, that is $X^T Y, X^T Z$ and $Y^T Z$, the gram matrix of their own samples, that is $X^T X, Y^T Y$ and $Z^T Z$, the gram matrix of the input data and the random mask, which are $a^T Y, a^T Z$ and $b^T Y$, and the gram matrix of some of the random masks, more precisely $a^T b, a^T c$ and $b^T c$. They form an incomplete gram matrix $\widetilde{K} = D^T D$, where $D = [X, Y, Z, a, b, c]$ (see Figure III.5). Even if the number of features is known by $\mathscr{A}$ and the complete version $K = D^T D$ is available, one can come up with multiple matrices satisfying the gram matrix K. Assume that there is a rotation matrix $R \in \mathbb{R}^{N \times N}$ where $N = 2(n_a + n_b + n_c)$. Then, we can compute a matrix E such that $E = R^{-1} D$. Hence, we can express D as $D = RE$. Then, the computation of $K = D^T D$ becomes as follows:

$$\begin{aligned} K &= D^T D \\ &= (RE)^T (RE) \\ &= E^T R^T R E \\ &= E^T R^{-1} R E \\ &= E^T E \end{aligned} \tag{III.4}$$

due to the property of the rotation matrices, which is $R^{-1} = R^T$. Based on this observation, we can say that for every new rotation matrix $\Theta \in \mathbb{R}^{N \times N}$ there exists a new matrix $\beta = \Theta^{-1} D$ satisfying $K = \beta^T \beta$. Since Aguilera and Pérez-Aguila [96] demonstrated a way to generate rotation matrices for any dimension, one cannot obtain a unique matrix satisfying K. To be exact, one cannot recover D from $K = D^T D$, which implies that it is not possible to recover D from $\widetilde{K} = D^T D$ either. □

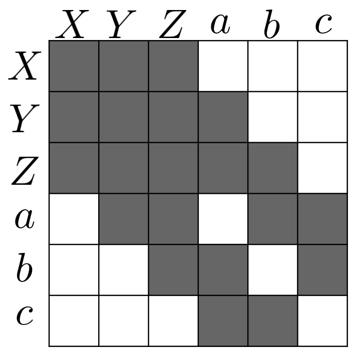


Figure III.5: The computed incomplete gram matrix in which only the shaded parts are available.

IV Privacy-preserving SVM on Outsourced Genomic Data via Secure Multi-party Computation

Huajie Chen   Ali Burak Ünal   Mete Akgün   Nico Pfeifer

Abstract

Machine learning methods are employed in many areas, such as medical data research, for their efficient and powerful data mining ability. However, submitting unprotected data to a third party, which attempts to train a machine learning model, may suffer from data leakage and privacy violation when the third party is compromised by an adversary. Hence, designing a protocol to execute encrypted computation is inevitably indispensable. In order to address this problem, we propose protocols based on secure multi-party computation to train a support vector machine model privately. Utilizing the semi-honest adversary model and oblivious transfer, the proposed protocols enable the training of a non-linear support vector machine on the combined data from various sources without sacrificing the privacy of individuals. The protocols are applied to train a support vector machine model with the radial basis function kernel on HIV sequence data to predict the efficacy of a certain antiviral drug, which only works if the viruses can only use the human CCR5 coreceptor for cell entry. Benchmarked on synthesized data with 10 data sources that consist of random generated integers, containing 100 labeled samples each, the protocol has consumed online time 2991.386/166.912 ms on average in arithmetic/boolean circuits, respectively. The cross-validation has reached 0.5819 F1-score on average on training data with the optimized parameters, which have reached 0.7058 F1-score afterwards on testing data set, which consist of protein sequence of CCR5 and its subtypes. The complete training and testing process on the real data, which contains in total 766 samples having 924 features after encoding, has consumed 43.75/15.84 seconds on average using arithmetic/boolean circuits, respectively, which shows the effectiveness and efficiency of our protocols compared to some of the existing studies in the literature.

IV.1 Introduction

Machine learning methods have been widely employed in many scientific research areas recently, solving problems by discovering patterns in the data. Applying machine learning methods, e.g. support vector machines (SVM), on medical data enables the detection of patterns which were previously unknown or too hard for a human entity to recognize. Hence it can be a contribution to medical technology development. For instance, novel HIV drug resistance mutations have been characterized by using clustering, multi-dimensional scaling and an SVM [97]. Network-based outcome prediction (NOP) method with the core of “sparse group lasso regression" has been programmed to predict the breast cancer state based on gene expression profiles [98]. Artificial neural network (ANNs) can yield prediction and prognosis of cancer [99].

With the latest technologies, such as second/third generation of DNA sequencing, obtaining full human genomes has become efficient and cheap, leading to many studies tackling a wide variety of health related research questions. However, the privacy of the patients can suffer from leakage during the process of accessing data by a third-party. To address this problem, privacy-preserving machine learning methods based on secure multi-party computation (MPC) [100, 101] and homomorphic encryption [102, 103, 104] were designed. However, all these approaches show different performances under different security assumptions. This means they have to make a trade-off between privacy and performance. To the best of our knowledge, efficient approaches utilizing MPC to train a non-linear SVM were lacking.

IV.1.1 Our Contributions

In this paper, we have designed and implemented an MPC-based solution for outsourcing SVM training to two non-colluding proxy servers. Our goal is to make privacy preserving training of an SVM on large volume data from multiple sources efficient. To this end, we propose two different solutions based on Beaver's multiplication triples [105] over arithmetic secret sharing and the Goldreich-Micali-Wigderson (GMW) protocol [26] over boolean secret sharing. We utilize single instruction multiple data (SIMD) operations which allows parallelization to increase the performance of our protocols.

In our protocols, the initialization phases can be completed locally. The genomic sequence data from various data sources, such as hospitals, are firstly encoded by employing one-hot encoding. Next, the encoded sequences are split into two shares and distributed onto computational parties, respectively. Self dot products are also locally computed, split and distributed onto the parties. In terms of training procedure, MPC is conducted between computational parties to get calculated results, which will be thereafter sent to the user and reconstructed to obtain the complete gram matrix which contains the dot product of the data from different sources as well as the dot product of the data from the same party. At the end, the user trains an SVM model on the desired kernel matrix which can be calculated by the derived gram matrix. Class labels are sent in plain text domain since they will not reveal any extra information regarding the samples.

We applied our protocol on HIV sequence data to predict the efficacy of a certain antiviral drug, which only works if the viruses can only use the human CCR5 coreceptor for cell entry. Our designed protocols based on arithmetic and boolean circuits have consumed 2991.38/166.912 ms online time executing circuits on synthesized data with 10 data sources that consist of random generated integers, containing 100 labeled samples each and in each sample there are 924 features, namely columns. We evaluated our results via F1-score and obtained 0.5819 on average on training data and 0.7058 on the test data, which certifies the potency of the protocol. The complete training and testing process on real data, which contains in total 766 samples having 924 features after encoding, took 43.75/15.84 seconds in arithmetic/boolean circuits protocol, respectively.

The rest of the paper is organized as follows: in Section IV.2, related works are presented. Required preliminaries and the concepts of our protocols in details are listed in Section IV.3 and IV.4, respectively. The evaluation of the protocols and the further discussion are given in Section IV.5. The paper is concluded in Section IV.6.

IV.2 Related Work

In the work of Yu et al.[106], a protocol is proposed to train an SVM model on the vertically partitioned data securely. In linear kernel, the multiplication of a matrix and its transpose will produce a gram matrix, which can also be generated by the addition of vertically partitioned matrix multiplication. Thereby, a random generated matrix of the same size as its local matrix will firstly be produced and passed to the next server iteratively until the last server passes the very last addition result back to the master server. By substracting the initial random matrix, the kernel matrix can thus be derived. Each server involved in this process can hence not breach into the real data. Training an SVM model on the resulting kernel matrix can then be done locally on the master server.

Vaidya et al. [107] have proposed a protocol to train an SVM model on the distributed data. In order to compute the gram matrix, the modified secure dot product calculation method is employed to protect the private information of the samples. The method is applicable on the vertically, horizontally and even arbitrarilly partitioned data. However, their method is mainly focus on the binary data.

In the work of Zhang et al. [108], they have addressed the privacy-preserving SVM training problem by avoiding utilizing fully homomorphic encryption (FHE) and partially homomorphic encryption and realizing proxy re-encryption on only one server. By employing the key switching technique, the designed protocol is able to compute dot product within one single cloud system, supporting further operations for SVM. However, the process is not so efficient on the large dimensional dataset due to the employed encryption technique.

In [100], Mohassel et al. have implemented a secure two-party computation (2PC) protocol in C++ language to realize privacy preserving machine learning for linear regression, logistic regression and neural network training. In the protocol, fixed-point addition and multiplication are two kernel operations. Shared decimal number can be manipulated. A specially designed activation function replacing the original rectified linear unit (ReLU) function has been brought out in this work to mitigate the problem incurred by the expensive computing time of MPC. Sharing switching is also employed to minimize the rounds of interaction and the number of oblivious transfer (OT). Vectorization, a concept raised in this paper, meaning operating matrices and vectors has been implemented based on linearly homomorphic encryption (LHE) and oblivious transfer and has achieved great improvement.

Wagh et al. [109] have developed a set of protocols using secure 3-party computation (3PC) for neural network training. They have made improvement in three ways, namely scalability, performance, and security. A new secure and efficient non-linear functions for linear layer, convolution, ReLU, and etc. have been implemented in this protocol, enabling the full capability of neural network training. The use of Yao's garbled circuits [110] is avoided due to the multiplicative overhead proportional to the safety parameter. Following this protocol honestly, none of the subsets of the servers can access the data and not a single malicious server can learn anything about the input and the output.

Ünal et al. [111] have proposed a framework to train an SVM model for a scenario where there are two parties having the data and one party that wants to perform training the model by using the data of those parties. They utilized randomized encoding, which means briefly that a function is encoded by random values, to provide security while training the model on genomic data set consisting of V3 loop sequences of HIV. The disadvantage of their approach is that the parties with the data should be active while the third party wants to have the gram matrix of those data.

There are many studies on privacy-preserving machine learning in the literature. Interested readers are referred to review by Tanuwidjaja et al. [112].

IV.3 Preliminaries

In this section, the utilized methods and algorithms underlying our approach are given.

IV.3.1 Sequence One-Hot Encoding

At data pre-processing stage, the sequences we utilized are encoded into numeric form by employing one-hot encoding. Let c_i be a character at the position i in the sequence S where $i \in \mathbb{Z}, i < |S|$. Since there are 20 amino acid residues and 1 gap, c_i can be encoded into a 21-digit form.

For example, if c_i is an alanine, c_i will be encoded into "1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0". If c_i is a gap, c_i is then "0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1". As the encoding program iterates, the whole sequence will be encoded into the aforementioned format.

IV.3.2 Measurement of Sequence Similarity

In order to assess the similarity of the sequences, the reverse of the Hamming distance is utilized. Hamming distance is described as the number of pairs of different characters at same positions between two strings having the same length. For instance, given two strings s_1 and s_2 of same length l, there are in total l characters in each string. The Hamming distance $D_{\mathbb{H}}$ is defined for the strings s_1 and s_2 as follows:

$$D_{\mathbb{H}}(s_1, s_2) = \sum_{i=1}^{l} I(s_1(i), s_2(i))$$

where l is the length of the sequences and I is the function which returns 1 if the given two characters, $s_1(i)$ and $s_2(i)$, are the same, 0 otherwise. In this project, the base similarity of two sequences are measured adversely compared to Hamming distance, that is $l - D_{\mathbb{H}}(s_1, s_2)$.

Based on Section IV.3.1, one sequence can now be considered as one vector $\vec{v}$ of binary values. If any two of the characters from any two sequences at the same position are identical, these two encoded characters will yield 1 as the result of the dot product, that is $\vec{v_1} \cdot \vec{v_2} = 1$, 0 otherwise. In this case, the more similarities two sequences share, the higher the dot product will be. Once the similarities of all possible pairs of sequences are computed, a matrix $\mathbf{M} \in \mathbb{Z}^{n \times n}$ where n stands for the number of sequences involved in the computation is obtained.

IV.3.3 Secure Multi-party Computation

Secure multi-party computaiton (MPC) was proposed in [110, 26] in early 1980s. Andrew Yao used MPC to address the famous "Millionaire Problem". MPC allows multiple parties, namely $P_1, \cdots, P_n$, to compute a function f on their private inputs $X = \{x_1, \cdots, x_n\}$ without revealing the private inputs to each other. At the end, each party learns the output of the function $f(X)$.

There are two types of adversaries leading to different security goals against the security of MPC protocols. Semi-honest adversaries are not allowed to abort the protocols but assumed to have the potential to learn additional information from the messages derived from the protocol execution, whereas the malicious adversaries are supposed to have the ability to completely deviate from the protocol[113]. Thereby, the mediator data will not leak the original information to any one of the two parties if the protocols are strictly followed.

Over the years, MPC has been speculated to be an impractical solution. However, with hardware improvements and optimizations, the applications based on MPC have become much more practical [114]. In this context, several studies have been presented to replenish MPC applications. For example, Demmler et al. [113] have designed a mixed-protocol framework that enables users to construct protocols using C++ language and converting share types freely. Pattuk et al. [115] have proposed a framework combining mixed protocol which is able to invoke the cheapest MPC cloud cost. In [116], a protocol utilizing additively homomorphic encryption and Yao's garbled circuits has been implemented in TASTY framework and has been adopted in face recognition applications.

Oblivious Transfer
Oblivious transfer [117] is an important building block of MPC. The sender employs the 1-out-of-2 OT to send one of the messages (m_0, m_1) to receiver based on the choice $c \in \{0,1\}$ of it. In the end of the protocol, the sender only learns m_c and the receiver does not learn c.

Notations
P_i denotes the computing party where $i \in \{0,1\}$. A shared value of x assigned to P_i is denoted by $\langle x\rangle_i^t$ where $t \in \{A: \text{Arithmetic}, B: \text{Boolean}\}$. $Rec_i^t(\langle x\rangle^t)$ represents the operator of reconstruction function where $x = Rec_i^t(\langle x\rangle^t)$.

Arithmetic Sharing
In arithmetic sharing, secret values are shared between two parties using additive secret sharing.

- **Shared Values:** Given arithmetic share $\langle x\rangle^A$ of bit length l, $(\langle x\rangle_0^A + \langle x\rangle_1^A) \mod 2^l = x$, $\langle x\rangle_0^A, \langle x\rangle_1^A \in \mathbb{Z}_{2^l}$.
- **Sharing:** Randomly generate $\langle x\rangle_i^A \in \mathbb{Z}_{2l}$ in P_i and send $x + 2^l - \langle x\rangle_i^A$ to P_{1-i} as $\langle x\rangle_{1-i}^A$.
- **Reconstruction:** $Rec_i^A(x) = \langle x\rangle_0^A + \langle x\rangle_1^A \mod 2^l$. $\langle x\rangle_{1-i}^A$ is sent to P_i.
- **Addition:** $\langle z\rangle^A = \langle x\rangle^A + \langle y\rangle^A$, $\langle z\rangle_i^A = \langle x\rangle_i^A + \langle y\rangle_i^A$ is computed locally on each P_i.
- **Multiplication:** $\langle z\rangle^A = \langle x\rangle^A \cdot \langle y\rangle^A$. Multiplication triplets $\langle c\rangle^A = \langle a\rangle^A \cdot \langle b\rangle^A$ are at first generated. In P_i, $\langle e\rangle_i^A = \langle x\rangle_i^A - \langle a\rangle_i^A$, $\langle f\rangle_i^A = \langle y\rangle_i^A - \langle b\rangle_i^A$. $Rec^A(e)$ and $Rec^A(f)$ are then performed in both party. $\langle z\rangle_i^A = i \cdot e \cdot f + f \cdot \langle a\rangle_i^A + e \cdot \langle b\rangle_i^A + \langle c\rangle_i^A$ is eventually computed on P_i.

Boolean Sharing
In boolean sharing, XOR based secret sharing is used to share secret values between two parties.

- **Shared Values:** Given a boolean share $\langle x\rangle^B$ of a bit, $x = \langle x\rangle_0^B \oplus \langle x\rangle_1^B$. "$\oplus$" denotes the operator of **XOR**.
- **Sharing:** Randomly generate an $r \in \{0,1\}$. Let $\langle x\rangle_i^B = r$ in P_i and send $x \oplus \langle x\rangle_i^B$ to P_{1-i} as $\langle x\rangle_{1-i}^B$.
- **Reconstruction:** $Rec_i^B(x) = \langle x\rangle_0^B \oplus \langle x\rangle_1^B$, $\langle x\rangle_{1-i}^B$ is sent to P_i.
- **XOR:** $\langle z\rangle^B = \langle x\rangle^B \oplus \langle y\rangle^B$. $\langle z\rangle_i^B = \langle x\rangle_i^B \oplus \langle y\rangle_i^B$ is computed locally on each P_i.
- **AND:** $\langle z\rangle^B = \langle x\rangle^B \wedge \langle y\rangle^B$. Boolean multiplication triplets are firstly generated: $\langle c\rangle^B = \langle a\rangle^B \wedge \langle b\rangle^B$. On each P_i, $\langle e\rangle_i^B = \langle a\rangle_i^B \oplus \langle x\rangle_i^B$ and $\langle f\rangle_i^B = \langle b\rangle^B \oplus \langle y\rangle^B$ are then computed. Afterwards, $Rec^B(e)$ and $Rec^B(f)$ are performed on each party. $\langle z\rangle_i^B = i \cdot e \cdot f \oplus f \cdot \langle a\rangle_i^B \oplus e \cdot \langle b\rangle_i^B \oplus \langle c\rangle_i^B$ is eventually computed on P_i.

IV.3.4 Support Vector Machine

As a supervised machine learning method, SVMs are frequently employed to solve classification problems. They have been shown to outcompete many other methods for low to medium size prediction problems, especially in combination with kernels. The optimization problem of SVM is formulated as follows:

$$\begin{aligned} \min_{\omega\in\mathbb{R}^d, b\in\mathbb{R}, \xi\in\mathbb{R}^n} \quad & \frac{1}{2}||\omega||^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & Y_i(\langle\omega, X_i\rangle + b) \geq 1 - \xi_i \\ & \forall\ i = 1, ..., n,\ \xi_i \geq 0 \end{aligned} \tag{IV.1}$$

where ω, b, ξ, C, n denote a vector with d dimensions, a parameter interfering the dot product between ω and X_i, a parameter determining how far a point can go beyond the hyperplane, the weight of punishment, and number of data points Y, following the listed order. X_i and Y_i represent the data features vector and its label. In the SVM model above, data points are allowed to cross the hyperplane but will be punished by the slack variables ξ_i scaled by the penalty factor C. By adjusting C, one can alter the weight of the punishment.

IV.3.5 Radial Basis Function Kernel

RBF kernel is commonly employed in SVM classification for its efficiency and universality [118].The kernel function is defined as follows:

$$\begin{aligned} K(\vec{x}, \vec{y}) &= \exp\left(-\frac{||\vec{x} - \vec{y}||^2}{2\sigma^2}\right) \\ &= \exp\left(-\frac{\vec{x}\cdot\vec{x} - 2\vec{x}\cdot\vec{y} + \vec{y}\cdot\vec{y}}{2\sigma^2}\right) \end{aligned} \tag{IV.2}$$

In this equation,“$\cdot$" represents the operator of dot product between vectors, whereas σ plays the role as the parameter adjusting the similarity level. If σ increases, the result of the exponential function will tend to be closer to 1, indicating that the similarity of two sequences is high, and vice versa. Furthermore, the second line of the equation indicates that the RBF kernel can be computed based on only the dot product of the vectors which we exploit to compute the RBF kernel matrix in our protocols.

IV.4 Privacy-Preserving SVM

In this section, we briefly describe our privacy-preserving SVM protocol systematically. In Section IV.4.1, we give an overview of it, afterwards each phase of the protocol will be illustrated in detail. The notations in Section IV.3.3 are succeeded here.

IV.4.1 System Overview

In our protocol, the genomic dataset from multiple medical institutions are outsourced to two non-colluding proxy servers in a secret-shared form. After getting request from a third-party who is allowed to query the data, such as a researcher, the two non-colluding proxy servers perform MPC to produce the required gram matrix of the samples for SVM training. The protocol structure is depicted in Figure IV.1. In this protocol, following parties are playing roles:

1. **Data Sources:** In this context, they are medical institutions, which are denoted as H_j, where $j \in \mathbb{N}^+$ and $j \leq n$, where n denotes the number of data sources. They upload their genomic data to the two proxy servers in a secret-shared form.
2. **User:** User U, such as a scientific researcher, who requests the two proxy servers to perform some secure computations over the secret shared data to produce the gram matrix for SVM training.
3. **Two Proxy Servers:** Two non-colluding servers are serving as computational parties, which are denoted as P_i, where $i \in \{0,1\}$, and they are responsible of the secure computation of the gram matrix. Selecting two semi-trusted third parties is believed to be the most practical and affordable solution [119] to secure the privacy preservation while utilizing genomic data to train a machine learning model.

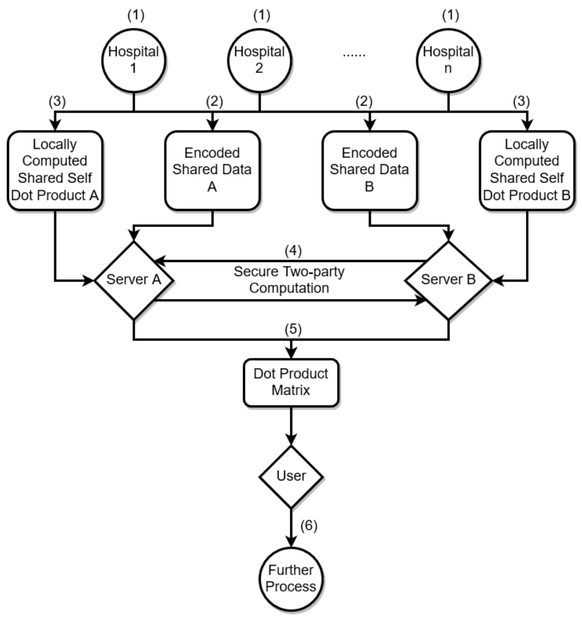


Figure IV.1: Protocol Structure: From Data Sources to MPC and Eventually to User's Process

Initialization

(1) Original sequence data in H_j are locally encoded into digital vectors (see Section IV.4.2).

(2) Encoded sequence data are split into two shares A/B locally in H_j and sent to P_0 and P_1, respectively (see section IV.4.3).

(3) Self dot products are pre-computed locally in H_j, afterwards they are also split into two shares A/B and sent to P_0 and P_1, respectively (see Section IV.4.4).

Training Procedure

(4) After receiving the query from U, MPC is conducted between P_0 and P_1 to get the desired processed data between files containing encoded sequences (see Section IV.4.5 and IV.4.6).

(5) Self and cross dot products are then integrated to obtain the whole gram matrix and passed to U (see Section IV.4.7).

(6) The gram matrix is utilized in U to train an SVM model and test the resulting model (see Section IV.4.8).

IV.4.2 Protein Sequence Encoding

The sequence encoding function takes one character that represents one residue and encodes it into a 21-digit format as described in Section IV.3.1. The output file contains the encoded sequences, where each row stands for one protein sequence and each of the 21 columns for one residue (or a gap).

IV.4.3 Secret Sharing of Encoded Data

The original encoded data is split into two shares using arithmetic sharing and boolean sharing. In arithmetic sharing based protocol, random integers less than 2^l are generated. Thereafter, the random integers will serve as share and help to construct the other share. In the booelan sharing based protocol, random bits are generated as one share and they are used to construct the other share. Eventually, in both protocols, two shares will be written into two files, respectively. Secret sharing of encoded data using arithmetic sharing is illustrated in Figure IV.2.

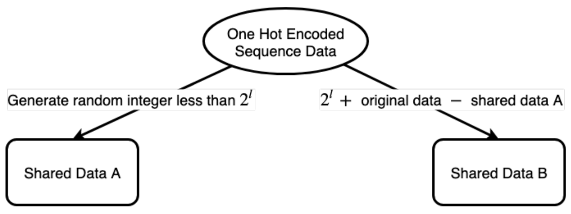

One Hot Encoded Sequence Data
Generate random integer less than 2^l
2^l + original data − shared data A
Shared Data A
Shared Data B

Figure IV.2: Shared Data Generation

IV.4.4 Self Dot Product Local Computation

Data sources compute the "self dot product", which means that the dot products of two instances of each data point are computed locally. Such local computation reduces the total communication cost. The outcomes of this process are two shares of the dot product matrix that are produced by the previously mention shared data generation technique. The shares of the dot product matrix are next distributed onto two third-party proxy servers, namely P_0 and P_1, and these shares can afterwards be reconstructed on U.

IV.4.5 Dot Product Computation using Arithmetic Circuits

All files that need multiplying into two long arrays are integrated, omitting unnecessary procedures to repeat the SIMD data generation. For example, given three files File A, B, C, which are denoted as f_a, f_b,and f_c, there are 3, 2, and 1 shared encoded sequences respectively contained in these files, which are represented by a_i, b_i, c_i, where i denotes the index of sequence in a certain file. The long array A will be "$a_1 a_2 a_3 a_1 a_2 a_3 a_1 a_2 a_3 b_1 b_2$", whereas the other long array will be "$b_1 b_1 b_1 b_2 b_2 b_2 c_1 c_1 c_1 c_1 c_1$". These operations are called GetLongArrayA() and GetLongArrayB(), respectively

```
1  Algorithm ComputeCrossDotProduct()
       input  : $sdf$: Shared data files, $dim$: Number of dimensions of a vector contained in file.
       output : $\langle arr_D \rangle_i^A$: Array containing dot product between files in shared form on $P_i$.
2      $\langle arr_A \rangle_i^A$ = GetLongArrayA($sdf$)
3      $\langle arr_B \rangle_i^A$ = GetLongArrayB($sdf$)
4      $\langle arr_C \rangle_i^A = \langle arr_A \rangle_i^A \cdot \langle arr_B \rangle_i^A$        ▷ Secure multiplication
5      $\langle arr_D \rangle_i^A$ = [] ▷ Create an empty array for summing up multiplication results in dot product blocks.
6      for $j$ in range($|\langle arr_C \rangle_i^A|$) do
7          $\langle sum \rangle = 0$
8          for $k$ in range($j$, $j + dim$) do
9              $\langle sum \rangle = \langle sum \rangle + \langle arr_C[k] \rangle_i^A$        ▷ Local addition
10         $\langle arr_D \rangle_i^A$.append($\langle sum \rangle$)
11     Output $\langle arr_D \rangle_i^A$
```

Algorithm IV.1: Dot Product Computation (Arithmetic Circuit)

Algorithm IV.1 construct two long arrays at first to reduce the times of creating SIMD data. By multiplying two long arrays in one time, the run time of the protocol has been successfully lowered. This algorithm omits the dot product computation between a file and itself for it can be previously locally calculated as described in Section IV.4.4. By putting an array into a SIMD share, operation on multiple data is thus enabled. After the multiplication, summation of multiplication results in each block of dot product computation is then executed, in order to locally sum up multiplication of each pair of elements from two vector separately. Eventually, the final result is loaded into an array. Figure IV.3 shows the general structure of the secure protocol based on arithmetic circuits.

IV.4.6 Dot Product Computation using Boolean Circuits

```
1  Algorithm ComputeCrossDotProduct()
       input  : $sdf$: Shared data files, $dim$: Number of dimensions of a vector contained in file.
       output : $\langle arr_D \rangle_i^B$: Array containing dot product between files in shared form on $P_i$.
2      $\langle arr_A \rangle_i^B$ = GetLongArrayA($sdf$)
3      $\langle arr_B \rangle_i^B$ = GetLongArrayB($sdf$)
4      $\langle arr_C \rangle_i^B = \langle arr_A \rangle_i^B \wedge \langle arr_B \rangle_i^B$        ▷ Secure AND
5      $\langle arr_D \rangle_i^B$ = [] ▷ Create an empty array for summing up multiplication results in dot product blocks.
6      for $j$ in range($|\langle arr_C \rangle_i^B|$) do
7          $\langle sum \rangle = 0$
8          for $k$ in range($j$, $j + dim$) do
9              $\langle sum \rangle = \langle sum \rangle + \langle arr_C[k] \rangle_i^B$        ▷ Secure addition
10         $\langle arr_D \rangle_i^B$.append($\langle sum \rangle$)
11     Output $\langle arr_D \rangle_i^B$
```

Algorithm IV.2: Dot Product Computation (Boolean Circuit)

Algorithm IV.2 is similar to Algorithm IV.1. The structure of the protocol is depicted in Figure IV.3.

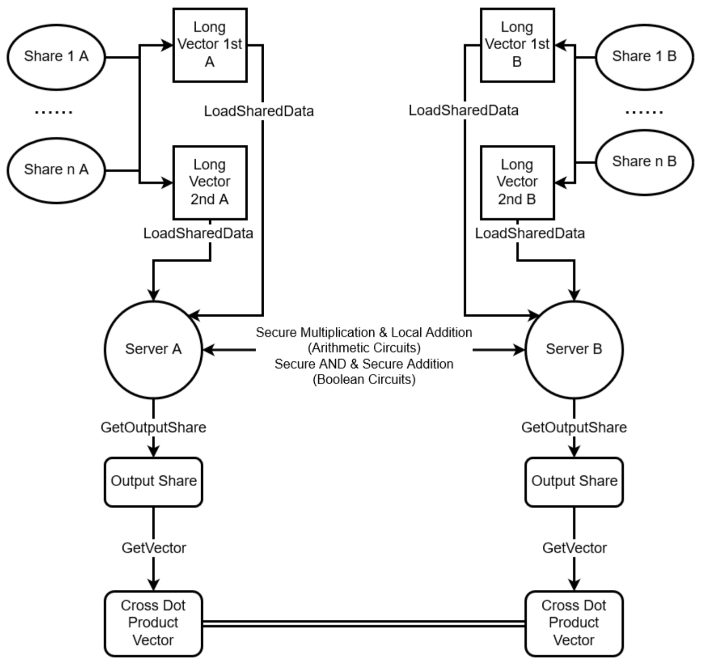


Figure IV.3: Dot Product Computation

The differences between these two are the types of shares and circuits. In this context, multiplication is executed between only 0 and 1, which fits the bit length (1-bit) of boolean shares. Therefore, secure AND operation in boolean circuits can replace secure multiplication in arithmetic circuits and accomplish better computation performance. The bit length of the shares can hence be reduced to 1, which can save a great deal of computational power during the communication process. In terms of boolean share, it varies much from arithmetic share and can thus not be summed up after multiplied in the previous way locally. A specific function based on secure addition is thereby designed for summing up the multiplication results in a dot product block.

IV.4.7 Result Array Integration & Dot Product Matrix Generation

Through our protocol, the self dot products can be reconstructed into one array on U manually. The index list of the access position in both self / cross dot product array will be then calculated. Based on the index list, a 2D vector, namely an $n \times n$ matrix, where n indicates the total number of sequences involved in the computation, will be generated and written into a file as the final output. In machine learning, the order of data and label must be strictly followed so that the model can be trained correctly. Therefore, the sorting process of the arrays is indispensable.

IV.4.8 SVM Cross Validation Experiment

Among all data, 20% of them are randomly chosen for testing the model. Once the gram matrix is computed, the parameters, which are the misclassification penalty parameter C and the class imbalance weight W of the SVM and the similarity adjustment parameter σ of RBF, are optimized by 5-fold cross-validation. C and σ are optimized over the set $S_C = [10^{-3}, 10^{-2}, \cdots, 10^{2}, 10^{3}]$, $S_\sigma =$

$[0.125, 0.25, 0.5, 1, 2, 4, 8]$, respectively, whereas the class weight is chosen from the set $S_W = [1:1, 1:2, 1:4, 1:8, 1:16]$, where former number indicates the class weight of label 0, which is the majority, and latter of label 1, which is the minority.The combination of parameters and the corresponding F1-score will be documented simultaneously. Eventually, the set of parameters achieving the highest F1-Score in the optimization process is determined. Afterwards, these optimal parameters are employed to train the final model. At the end, the trained model is tested on the test data and the predictions are evaluated by using F1-score.

IV.4.9 Security Analysis

Outsourcing data to N non-colluding semi-honest third parties was first studied in [120]. The authors also presented the security proof of the generic solution. In our protocols, utilizing the outsourcing idea in [120], multiple medical institutions outsource genomic data to two non-colluding semi-honest third parties. We need to protect the privacy of individuals whose genomic data are outsourced to the two non-colluding servers. The security of our protocols based on arithmetic circuits and boolean circuits depends on the proven security of the protocol based on Beaver's multiplication triples [105] and the GMW protocol [26] respectively. A semi-honest adversary compromising at most one proxy server can get one share of the private genomic data of the individuals. We know that genomic data is shared using arithmetic or boolean sharing so it looks like a uniformly random data. This ensures that the adversary can not obtain anything about private genomic data of individuals. Our protocols are secure against malicious data providers and users. Data providers only send genomic data to the two proxy servers in a shared form and do not receive any messages from other parties. Their malicious inputs do not result in any data leakage. Any changes in the input of the user result in different analysis response. This ensures security against malicious users. Confidentiality, integrity and authentication between all communicating parties are provided using state-of-the-art technologies such as TLS [121].

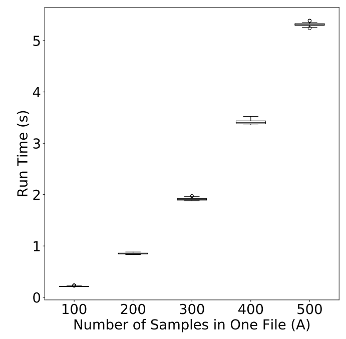
IV.5 Evaluation & Discussion

The GMW protocol [26] is used for operations on boolean circuits and the protocol based on Beaver's multiplication triples [105] is employed for operations on arithmetic circuits. Our protocols have been implemented using C++ and ABY framework [113]. ABY framework allows researchers to use combination of schemes based on arithmetic sharing, boolean sharing and Yao's garbled circuits. It supports three types of data shares and converts between any two types, which provides high degree of freedom in the protocol design.

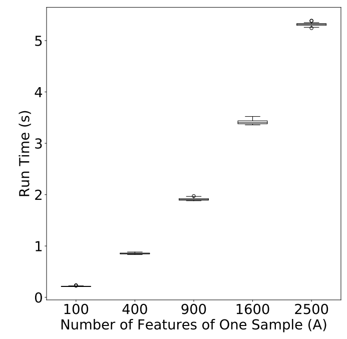
This work has been benchmarked and run on two identical computers with Intel Core i7-7700HQ equipped with 2.80GHz CPU and 16.0GB 2667MHz RAM connected via gigabit ethernet.

IV.5.1 Experiment on Synthesized Data

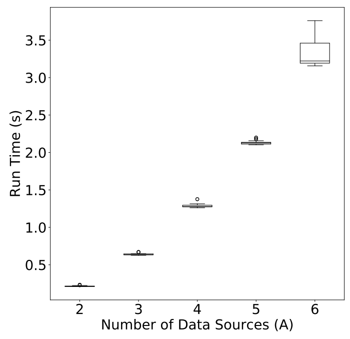
The online run time of the protocols consumed in performing the MPC on synthesized data has been documented. This process has been repeated for 25 times for each selected parameters. The relation between the online time and the number of sequences/dimensions/data sources has been depicted in Figure IV.4 and IV.5. Thereby, the positive correlation between the online time and the concerned parameter can be observed. Ascending respective parameters lead to increasing the online time of the protocols. In addition, the boolean circuits protocol has outperformed the arithmetic circuits protocol under each same situation.



(a) #Features=100, #Data Sources=2

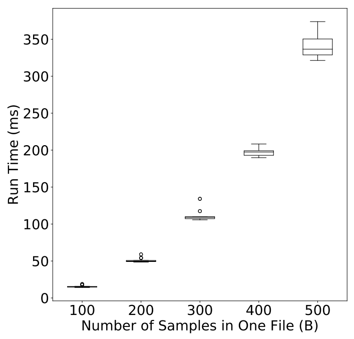


(b) #Samples=100, #Data Sources=2

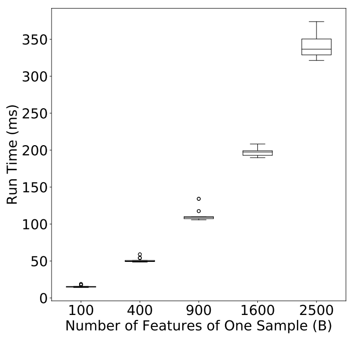


(c) #Samples=100, #Features=100

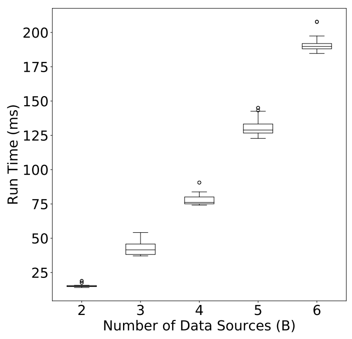
Figure IV.4: Online Time against Number of Samples/Features/Data Sources in Arithmetic (A) Circuits Protocols.



(a) #Features=100, #Data Sources=2



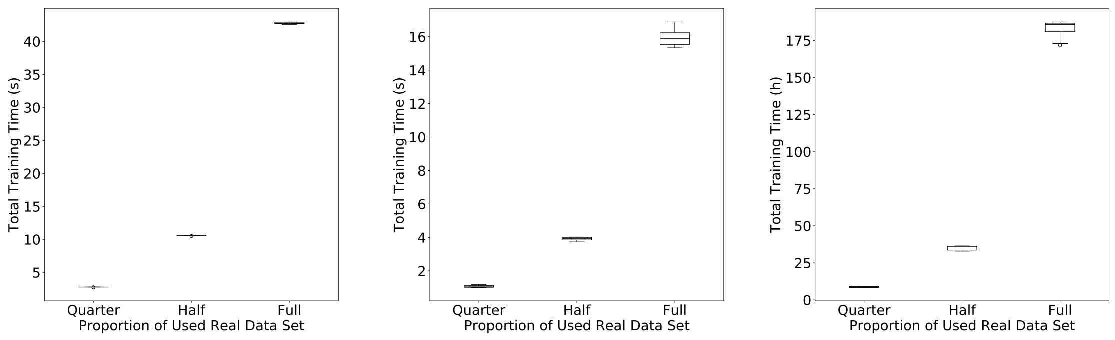
(b) #Samples=100, #Data Sources=2



(c) #Samples=100, #Features=100

Figure IV.5: Online Time against Number of Samples/Features/Data Sources in Boolean (B) Circuits Protocols.

With the help of the aforementioned "Long Array Integration", the time required to construct SIMD shares has been successfully lowered and the run time has been thus reduced. Nevertheless, the run time in arithmetic circuits still needs improving. The encoded sequence data consist of only 1 and 0, which means that the data can be bitwisely shared. Hence, we implemented a protocol based on boolean circuits by substituting the arithmetic circuits. Compared to the arithmetic shares, boolean shares have only 1 bit length and can only be manipulated bitwisely, whereas the arithmetic shares are of 8 bit length in this context, burdening the communication cost. Though the arithmetic/boolean circuits need to generate multiplication/and triples, respectively, while performing secure multiplication or secure AND, and the boolean circuits need extra online secure addition, the total protocol run time of the boolean circuits has still outperformed the other. The underlying reason for such difference between the boolean and arithmetic circuits is mainly because of the reduced communication data which stems from having less data to transfer in the boolean



(a) Arithmetic Circuits Protocol (b) Boolean Circuits Protocol (c) Key Switching

Figure IV.6: Run Time Comparison between Arithmetic/Boolean Circuits Protocol & Key Switching

circuits. Hence, the performance has been significantly enhanced using the protocol in boolean circuits.

IV.5.2 Experiment on Real Data

In Figure IV.6, the comparison between three different methods with respect to run time is shown. Three different approaches have been benchmarked on real experiment data of various proportion and repeated for 10 times, respectively. Our two protocols has significantly outperformed the key switching methods in the total SVM training time. The boolean circuits protocol remains still the best in terms of the execution time.

After the benchmarking process, the program has been tested on real data set consisting of 766 protein sequences. Those sequences are split into 2 files each of which contains 383 sequences, representing 2 sets of data originating from 2 hospitals. The sequence data was firstly encoded and partitioned into two shares for each. Next, the self dot product was calculated and also split into two shares separately. Once the shares are prepared, they were sent to the two servers. Thereafter, the cross dot product computation started based on the protocols. Eventually, the resulting gram matrix was utilized to compute the RBF kernel matrix via the formula given in Equation IV.2. Once the RBF kernel matrix is calculated, an SVM is trained by employing the resulting kernel matrix. The 5-fold cross validation has reached 0.5819 F1-score on average (standard deviation: 0.1509) with the optimized parameters. Using these parameters, the SVM accomplished 0.7058 F1-score on the test data set. A proper explanation to the higher F1-score in the testing process compared to the training process can be the bias of the data set. The standard deviation of the cross validation is 0.1509, indicating that the F1-score may happen to be significantly higher or lower than the average score. Thus, 0.7058 is achieved coincidentally. Due to the limitation of the RAM, the computation process must be split into 24 steps. In total, the arithmetic/boolean circuits protocol has consumed 43.75/15.84 seconds for the entire training and testing process on the full data set, respectively, with respect to the data originating from 2 data sources and having 388 respective samples and in each of the samples 924 features.

In terms of the F1-Score of the SVM model, unequal distribution of the labels in the training and test data sets may be the incurring reason to the relatively lower final result. One could create data sets having the same ratio of the labels to further improve the performance. In the meantime, a new model that enables better accuracy and data form that carries more information are thus demanded. Online time can be further decreased with larger bandwidth and better implemented protocols. Meanwhile, sufficient memory allows the computation to be done in less rounds, saving the communication cost. Inevitably, the cost of local setup time is immutable. In order to mitigate this issue, there is an option in our protocols allowing server to generate all needed parameters in anytime, making preparation for the online communication. Thereby, users can start getting access to the computation at any time with already generated parameters. The set of parameters will be discarded and replaced by the renewed ones as soon as they are used, ensuring the security of the MPC.

IV.6 Conclusion

In this paper, we presented two schemes that enable privacy-preserving training of an SVM on outsourced genomic data from multiple sources. In our schemes, we utilized the arithmetic secret sharing, the boolean secret sharing, and oblivious transfer to make SVM training secure and efficient simultaneously. Our protocols are executed on two semi-trusted proxy servers, each of which only accesses one share of the resulting dot product that is needed for the training of an SVM. Our schemes are secure under the semi-honest adversary model. We conducted experiments on both a real HIV data set and synthetic data sets in order to show the efficacy and efficiency of our protocols. For future work, we will make our protocols secure against malicious adversaries without decreasing the performance noticeably.

Acknowledgement

This study is supported by the DFG Cluster of Excellence “Machine Learning – New Perspectives for Science”, EXC 2064/1, project number 390727645. Furthermore, NP and MA acknowledge funding from the German Federal Ministry of Education and Research (BMBF) within the ‘Medical Informatics Initiative’ (DIFUTURE, reference number 01ZZ1804D).

V CECILIA: Comprehensive Secure Machine Learning Framework

Ali Burak Ünal **Mete Akgün** **Nico Pfeifer**

Abstract

Since machine learning algorithms have proven their success in data mining tasks, the data with sensitive information enforce privacy preserving machine learning algorithms to emerge. Moreover, the increase in the number of data sources and the high computational power required by those algorithms force individuals to outsource the training and/or the inference of a machine learning model to the clouds providing such services. To address this dilemma, we propose a secure 3-party computation framework, CECILIA, offering privacy preserving building blocks to enable more complex operations privately. Among those building blocks, we have two novel methods, which are the exact exponential of a public base raised to the power of a secret value and the inverse square root of a secret Gram matrix. We employ CECILIA to realize the private inference on pre-trained recurrent kernel networks, which require more complex operations than other deep neural networks such as convolutional neural networks, on the structural classification of proteins as the first study ever accomplishing the privacy preserving inference on recurrent kernel networks. The results demonstrate that we perform the exact and fully private exponential computation, which is done by approximation in the literature so far. Moreover, we can also perform the exact inverse square root of a secret Gram matrix computation up to a certain privacy level, which has not been address in the literature at all. We also analyze the scalability of CECILIA to various settings on a synthetic dataset. The framework shows a great promise to make other machine learning algorithms as well as further computations privately computable by the building blocks of the framework.

V.1 Introduction

In recent years, machine learning algorithms, especially deep learning algorithms, have become more sophisticated and are now more data-driven to achieve better performance. At the same time, the number of sources generating data with sensitive information and the amount of that data have increased dramatically. By its very nature, the privacy of this data must be protected during collaborative model training and testing. Likewise, the privacy of such a model must be kept private when the owner of the trained model deploys it as a prediction service.

In order to preserve the privacy of both the data and the model, there are several privacy techniques utilized in the literature. One of these techniques is differential privacy (DiP) introduced by Dwork et al. [27]. In DiP, to protect the privacy of the components involved in the process, one needs to perturb the input data, the model parameters, and/or the output by adding noise in a certain budget that adjusts the privacy level[122, 123]. By adding noise, one has to sacrifice to some extent the performance of the model and exactness of the result. In addition to DiP, there is a cryptographic technique called homomorphic encryption (HE) that takes into account the missing points of DiP in addition to the basic privacy requirements. HE protects the privacy of the data and/or the model by encrypting the data and/or the parameters of the model with different key schemes. Thanks to this mechanism, various operations such as addition and multiplication can be performed in the encrypted domain. There are several attempts to implement machine learning algorithms using HE [35, 124, 125]. However, the drawbacks of HE are its huge runtime and the limited number of practical operations that can be realized with HE. Secure multi-party computation (MPC), on the other hand, satisfies these requirements in addition to those already mentioned. The idea is to employ several parties performing the required computations and to share the data and/or the model parameters among these parties in such a way that none of the parties can learn about the data and/or the model parameters on its own. There are several MPC frameworks proposed in the literature to address various machine learning algorithms [126, 127, 34, 128]. Although they have several efficient and secure basic functions used by convolutional and feedforward neural networks, from which we have also benefited, they lack the exact computation of more complex operations, such as the exponential computation and the inverse square root of a Gram matrix.

In this study, we introduce a general purpose efficient secure multi-party computation framework, CECILIA, based on 3 computing parties. In addition to the inherited or adapted methods such as addition, multiplication, the most significant bit and multiplexer, CECILIA provides two novel functions, to the best of our knowledge, for the first time, which are the privacy preserving exact computation of the exponential of a known base raised to the power of a secret shared value and the inverse square root of a secret shared Gram matrix. We show the potential of our framework CECILIA by realizing privacy preserving prediction on a specific type of pre-trained deep neural network, namely recurrent kernel networks [17], which has more complex operations than most of the other deep neural networks. As a summary, we can list the contributions of our paper:

- We present a new comprehensive secure machine learning framework providing basic building blocks addressing the functions used in machine learning algorithms.
- For the first time, CECILIA provides the exact privacy preserving computation of the exponential of a known base raised to the power of a secret shared value.
- CECILIA enables the privacy preserving inverse square root of a secret shared Gram matrix as the first attempt in the literature.
- We introduce the first efficient approach of privacy preserving inference on recurrent kernel networks via CECILIA.

V.2 Preliminaries

V.2.1 Security Model

In order to prove the security of CECILIA, we use the honest-but-curious security model, in other words we analyze the security of the framework when there is a semi-honest adversary who corrupts a party and follows the protocols honestly, but at the same time tries to infer information during the execution of these protocols. In our analyses of the protocols proposed in CECILIA, we consider a scenario where a semi-honest adversary corrupts either one of the proxies or the helper party and tries to infer the input and/or output values of the executed function. We take the same approach to analyze the security of privacy preserving RKN and focus on if the semi-honest adversary corrupting a party can break the privacy of the test sample of the data owner and/or the model parameters of the model owner during the prediction of a test sample using the outsourced model.

V.2.2 Square-and-multiply Algorithm for Exponential

One of the approaches to computing the exponential of a base raised to the power of value that can be represented in a binary form is the square-and-multiply algorithm. Let b be the base and a be the power, whose binary representation is $\langle a \rangle$ of length n. Then, one can compute b^a as follows:

$$b^a = \prod_{i=0}^{n-1} b^{\langle a \rangle_i \cdot 2^i} \tag{V.1}$$

where $\langle a \rangle_i$ represents the bit value of a at position i, assuming that the indexing starts from the least significant bit, that is $\langle a \rangle_0$ corresponds to the least significant bit and $\langle a \rangle_{n-1}$ corresponds to the most significant bit.

V.2.3 Recurrent Kernel Networks

Chen et al. [17] gave a kernel perspective of RNNs by showing that the computation of the specific construction of RNNs mimics the substring kernel allowing mismatches and the local alignment kernel which are widely used on sequence data [18, 19, 20]. In short, they construct an RNN, which they call a recurrent kernel network (RKN), that performs kernel function computation by benefiting from the recursive structure of the similarity computation. In addition to a well-designed kernel formulation, this brings the advantage that the parameters of the model can be optimized via backpropagation, allowing the RKN to outperform the traditional substring kernel and the local alignment kernel as well as the LSTM [22]. Such performance promises to have a large impact on the results of tasks with small to medium size data.

In RKN, small motifs, called *anchor points*, are used as templates to measure similarities among sequences. The anchor point encoding is similar to one-hot encoding with one small difference. The encoding of each character of these anchor points does not indicate what that character is exactly, but rather the probability that that character is a certain single character in the alphabet. Let there be q anchor points of length k, each of whose characters is encoded using a vector of size d, and a sequence x of length s, whose characters are encoded using one-hot encoded vectors of length d. For the t-th character of the input sequence, the RKN approach first computes the similarity of this character to each character of all anchor points by the following:

$$K(x_t, z_j^i) = e^{\alpha(\langle x_t, z_j^i \rangle - 1)} \tag{V.2}$$

where α is the similarity measure parameter, x_t is the d-dimensional vector representation of the t-th character of the sequence x and z_j^i is the d-dimensional vector representation of the i-th character of the j-th anchor point. Once the similarity of the t-th character of the sequence to the j-th character of all q anchor points is computed and represented as a vector $b_j[t]$, the computation continues as follows:

$$c_j[t] = \lambda c_j[t-1] + c_{j-1}[t-1] \otimes b_j[t] \tag{V.3}$$

where $c_j[t]$ is the initial mapping of the sequence up to its t-th character into a q-dimensional vector based on anchor points of length j, λ is a scalar value to downgrade the effect of the previous time points for $j \in \{1,\ldots,k\}$ and $t \in \{1,\ldots,s\}$ and $\otimes$ is Hadamard product. As the base of this recursive equation, $c_0[t]$ is a vector of 1s and $c_j[0]$ is a vector of 0s if $j \neq 0$.

To obtain the final mapping of the sequence, the RKN approach multiplies $c_j[s]$ by the inverse square root matrix of the Gram matrix of the anchor points up to their j-th characters. Afterwards, in the substring kernel allowing mismatches, they use a linear layer, which is a dot product between $c_k[s]$ and the weight vector of the classifier w, to obtain the prediction of the input sequence. Figure V.1 depicts the computations of an RKN for a single character of the input sequence.

V.3 Framework

CECILIA is based on 3 computing parties. Two of them are called *proxy* and are the parties with which the external entities such as the model owner and the data sources interact. The third one is the *helper* party helping the proxies compute the desired function without breaking the privacy. It provides the proxies with the shares of the purposefully designed values or performs calculations on the data masked by the proxies in the real number domain and returns the share of these calculations. To ensure the privacy in CECILIA, we utilize 2-out-of-2 additive secret sharing. In this scheme, let $\langle x \rangle_i$ be the share of x represented with n-bits in party P_i for $i \in \{0,1\}$, one can reconstruct x by computing $(\langle x \rangle_0 + \langle x \rangle_1) \mod L$. In the rest of this section we describe the number format we use to represent values. We then introduce the methods provided by CECILIA.

V.3.1 Number Format

Since the numbers in the methods of CECILIA can be both positive and negative real numbers, we need to display the fractional part along with the integer part. To achieve this, we follow the number format proposed by Mohassel and Zhang [34] in SecureML. In this number format, a certain number of least significant bits represents the fractional part of the value and the rest expresses the integer part.

V.3.2 Addition (ADD)

The proxies add the shares of two secret shared values they have and obtain the share of the addition of these values without any communication.

V.3.3 Multiplication (MUL)

The base of the multiplication is the pre-computed multiplication triple [105]. However, because of the special number format, there is an additional operation called *truncation*. The goal of truncation is to preserve the number format after multiplication. Let $\langle c \rangle_i = \langle a \rangle_i \cdot \langle b \rangle_i$ be the shares of the multiplication triple in P_i for $i \in \{0,1\}$. To compute $\langle z \rangle = \langle x \rangle \cdot \langle y \rangle$, P_i first computes $\langle e \rangle_i = \langle x \rangle_i - \langle a \rangle_i$ and $\langle f \rangle_i = \langle y \rangle_i - \langle b \rangle_i$, and then sends them to P_{1-i}. Afterwards, P_i reconstructs e and f and calculates $\langle z \rangle_i = i \cdot e \cdot f + f \cdot \langle a \rangle_i + e \cdot \langle b \rangle_i + \langle c \rangle_i$. In the last step to obtain the shares of the multiplication, P_0 shifts z_0 to the right by f. Slightly differently, P_1 first shifts $(-1 * z_1)$ to the right by f and then multiplies the result by -1. For more details, see [34, 127].

V.3.4 Comparison of Two Secret Shared Values (CMP)

In order to compare two secret shared values in CECILIA, we utilize the comparison method proposed by Ünal et al. [129]. In CMP, the proxies input the secret shares of x and y, and receive 0 if $x \geq y$, 1 otherwise in secret shared form.

V.3.5 Multiplexer (MUX)

One of the common methods used by the proxies is the multiplexer. Due to privacy, the proxies must choose between two options without knowing which one is chosen. To fulfill this requirement, we adapt the multiplexer method [129] to CECILIA. The proxies input the secret shares of x and y into MUX along with the secret shared selection bit s. Then, the proxies receive the fresh secret share of x if $s = 0$, y otherwise.

V.3.6 Most Significant Bit (MSB)

CECILIA deals with the private determination of the most significant bit of a secret shared value x by MSB [129]. Given x, MSB returns the secret shared form of the most significant bit of x. In the number format of CECILIA, 0 means that $x \geq 0$ and 1 means that $x < 0$.

V.3.7 Dot Product (DP)

Since the dot product between two vectors is a widely used method in machine learning algorithms, we provide DP in CECILIA. It essentially uses MUL to compute the element-wise multiplication of the vectors and sums the elements of the resulting vector to obtain the dot product result.

V.3.8 Exponential Computation (EXP)

One of the novel methods of CECILIA is the exponential of a publicly known base raised to the power of a given secret shared value. For this purpose, we have been inspired by the square-and-multiply algorithm. We extend the core idea of this algorithm to cover not only the positive numbers, but also the negative numbers as well as their decimal parts in a multi-party scenario. Briefly, we compute the contribution of each bit to the result in advance and privately select the correct contribution depending on the sign of the power and the bit value corresponding to each contribution. Then we privately multiply the selected contribution to obtain the share of the exponentiation.

More precisely, at the beginning, the proxies calculate the possible contribution of each bit of the positive and negative real numbers to the result of the exponentiation. That is, they calculate the exponential of the base as if only that bit of the power is 1 and the rest is 0. Here, it is important to note that the proxies take into account the binary representation of the absolute value of the negative power throughout the exponentiation. For positive powers represented in our number format, the integer part is the same as the original square-and-multiply algorithm. However, for the decimal part, they change the update operation of the base from taking the square of the base to the square root of the base. This leads the base to approach 1 when one proceeds further in the decimal part of the power. When calculating the possible contribution of a negative real number in the power, the proxies follow the same procedure as for the positive power, with the only difference in the starting value of the base. Instead of starting with the original base, they start with the reciprocal of the base, that is 1 divided by the original base. Note that one can compute these contributions once in offline time and reuse them later to save some time if the base is known in advance.

Once the proxies have calculated the possible contributions of the bits of positive and negative number in the power, they first use MSB to determine the sign of the power. Then they use the output of MSB in MUX to privately select between these contributions without knowing which one is selected. They also use this bit to select the power itself or the result of subtracting the power from 0. If the power is positive, they select the power itself. If it is negative, they select its absolute value without knowing which one is selected. After this point, we refer to the output of this selection as power, regardless of the selection.

After selecting the power and the corresponding set of possible contributions, they determine the secret shared values of each bit of the power using MSB. They first call MSB and then shift the share of the power by 1 to the left and repeat these steps until they have processed all bits. Afterwards, they input the resulting secret shares of each bit of the power into MUX as selection bits to choose between the contributions of the power selected based on the sign and the vector of plain 1s, which indicates the contribution of the bits if the bits are 0. Once they have the result of the MUX operation, they use MUL operations to multiply all the selected contributions to obtain the result of the exponentiation.

V.3.9 Inverse Square Root of Gram Matrix (INVSQRT)

In addition to the usual operations, CECILIA also provides a more sophisticated and special operation, namely the inverse square root of a secret shared Gram matrix (INVSQRT). To compute the inverse square root of a secret shared Gram matrix, we follow the idea of computing the reciprocal of the square root of the eigenvalues of the Gram matrix and reconstructing the desired matrix using the diagonal matrix of these reciprocals of the square root of the eigenvalues and the original eigenvectors. The first step, then, is to perform a private eigenvalue decomposition of the Gram matrix. For this purpose, we were inspired by the approach proposed by Zhou and Li [42], where the eigenvalue decomposition was designed as an outsourcing operation from a single party. We adapted this approach to the multi-party scenario such that the parties own a share of the Gram matrix and perform the eigenvalue decomposition without learning anything other than their share of the outcome.

Let G_i be the share of the Gram matrix in P_i for $i \in \{0,1\}$, $\mathscr{M}$ and (τ, s) be a common orthogonal matrix and common scalars, respectively, known by both proxies. The proxies first mask their share of the Gram matrix by computing $G_i^{'} = \mathscr{M}(\tau G_i + sI)\mathscr{M}^T$, where I is the identity matrix, and then send it to the helper party. First, the helper party reconstructs $G^{'}$ and then performs an eigenvalue decomposition on $G^{'}$. It obtains the masked versions of the original eigenvalues, represented as $\Lambda^{'}$, and eigenvectors, represented as $Q^{'}$. The helper party splits $Q^{'}$ into two secret shares and sends them to the corresponding proxies. To unmask $Q_i^{'}$ and obtain the share of the eigenvectors Q of G, the proxies compute $\mathscr{M}^T Q_i^{'}$. As for the eigenvalues, the helper party first generates a vector of random values Δ and a random scalar α, and sends them to P_1. Once P_1 has received Δ and α, it computes the vector $U = s\Delta + \alpha$, which will act as *unmasker*, and sends it to P_0. The final task of the helper party is to mask the masked eigenvalues by computing $\Lambda'' = \Lambda' \odot \Delta + \alpha$ and to send it to P_0. Once P_0 receives U and Λ'', it computes $\Lambda''' = (\Lambda'' - U)/\tau$. To obtain the shares of the reciprocal of the square root of the original eigenvalues, i.e. $\Lambda^{-1/2}$, P_0 and P_1 perform private multiplication with inputs $((\Lambda''')^{-1/2}, 0)$ and $(0, \Delta^{1/2})$, respectively. The final step for the proxies to obtain the share of the inverse square root of G is to reconstruct it by computing $G_i^{-1/2} = Q_i \cdot diag(\Lambda^{-1/2}) \cdot Q_i^T$ for $i\ in \{0,1\}$.

It is important to note that INVSQRT is a private method with a high probability, since $\mathscr{M}$, τ and s must be in a problem specific range so that $G^{'}$ does not overflow and lose its connection to the real numbers. This restriction affects the feature of complete randomness of the shares in secure multi-party computation. Based on the resulting values from the computation, the parties can reduce the range of secret values, which in this case are the values in G, Q and Λ, to a range slightly smaller than the original range. However, in order to perform private inference with RKN, one can outsource the inverse square root of the required Gram matrices along with the other parameters of the model so that the randomness regarding the aforementioned values is preserved. The reason why we have included INVSQRT here is to initiate the effort of realizing such a computation, which is required if one wishes to train an RKN or other algorithms requiring this operation from scratch in a secure multi-party computation. We discuss further details in Section V.5.

V.4 Privacy Preserving RKN (ppRKN)

In this section, we explain the procedure of the privacy preserving inference on RKN, which is called ppRKN.

V.4.1 Outsourcing

The first step for the private inference is the outsourcing of the model parameters from the model owner and the test sample from the data owner to the proxies. To outsource the model parameters, which are the anchor point matrix, the linear classifier weights and the inverse square root of the Gram matrices of the anchor points, if one does not want to use INVSQRT for the sake of fully randomness, the model owner splits them into two arithmetic shares and sends them to the proxies in such a way that each proxy has a single secret share of each parameter. In the outsourcing of the test samples, the data owner proceeds similarly after using one-hot encoding to convert a sequence into a vector of numbers. It divides this vector into two shares and sends them to the proxies.

V.4.2 Private Inference

After outsourcing the model parameters and the test sample, we use the building blocks of CECILIA to realize the private inference on pre-trained RKN. Let $t \in \{1, \dots, s\}$ be the index of the characters in the sequence. The first step is to compute the similarity of the one-hot encoded t-th character of the sequence to each character of each anchor point via Equation V.2. Such a similarity calculation involves the dot product of two secret shared vectors, the subtraction of a plaintext scalar value from a secret shared value, the multiplication of a plaintext scalar value by a secret shared value and the exponential of a known base raised to the power of a secret shared value, respectively. The output of this process corresponds to the component $b_j[t]$ in Equation V.3. Once the similarity computation is complete, the proxies proceed with the private elemenent-wise product between

$b_j[t]$ and $c_{j-1}[t-1]$, which is the initial mapping of the sequence up to the $(t-1)$-th character based on the anchor points of length $j-1$. Then the proxies add the result of the element-wise product with the downgraded $c_j[t-1]$ with a plaintext scalar value λ. At the end of this computation, the proxies obtain the secret shared initial mapping of the sequence, namely $c_j[t]$, up to the t-th character to q dimensional space based on each anchor point of length $j \in \{1,\ldots,k\}$.

After computing the initial mapping of the sequence up to its total length, that is obtaining $c_k[s]$, the proxies either compute the inverse square root of the Gram matrices of the anchor points up to each length of the anchor points via INVSQRT, which is $K_{Z_jZ_j}^{-1/2}$, where Z_j is the anchor points up to the j-th character, or use the inverse square root of the Gram matrices directly if they are also outsourced. To obtain the final mapping, they multiply these matrices by the corresponding initial mapping vectors of the sequence. Afterwards, the proxies perform the private dot product computation of two secret shared vectors, which are the weights of the classifier and the mapping of the sequence. In the end, they obtain the prediction of ppRKN for the given sequence in a secret shared form. These shares can then be sent back to the owner of the data enabling the reconstruction of the prediction of the ppRKN.

V.5 Security Analysis

In this section, we provide the security analysis of the functions of CECILIA that are novel and ppRKN. For the security analysis of the adapted or taken functions, we kindly refer the reader to the corresponding papers.

Lemma 9. The protocol EXP securely computes the exponential of a publicly known base raised to the power of a secret shared value.

Proof. The proof is in the Supplement. □

Lemma 10. The protocol INVSQRT privately computes the inverse square root of a secret shared Gram matrix with a high probability.

Proof. We first demonstrate the correctness of INVSQRT. Once the Gram matrix G whose eigenvalues and eigenvectors are Λ and Q, respectively, is masked by $G^{'} = \mathscr{M}(\tau G_i + sI)\mathscr{M}^T$, we conduct eigenvalue decomposition on $G^{'}$ resulting in the masked eigenvalues and eigenvectors of G, which are $\Lambda^{'} = \tau\Lambda + s$ and $Q^{'} = \mathscr{M}Q$, respectively. The recovery of Q can be achieved by $Q = \mathscr{M}^T Q^{'}$ since $M^T = M^{-1}$ due to the orthogonality. Since the masked eigenvalues $\Lambda^{'}$ are, once again, masked by computing $\Lambda^{''} = \Delta\Lambda^{'} + \alpha$, we have to compute the partial unmasker $U = s\Delta + \alpha$ by using Δ, α and s, and subtract it from $\Lambda^{''}$ and divide the result by τ to obtain $\Lambda^{'''} = \Delta\Lambda$. Then, the multiplication of $(\Lambda^{'''})^{-1/2}$ and $\Delta^{1/2}$ gives us the reciprocal of the square root of the original eigenvalues, i.e. $\Lambda^{-1/2}$. Finally, we construct the inverse square root of G by computing $Q \cdot diag(\Lambda^{-1/2}) \cdot Q^T$. This concludes the correctness of INVSQRT.

Regarding the security of INVSQRT, we adapted this method from the private eigenvalue decomposition provided by Zhou and Li [42]. Since they utilize the real number domain rather than a ring, one can employ brute-force attack to infer the values as they also pointed out. The protection of the privacy of the data, however, comes from the fact that the attack has non-polynomial bounds [42].

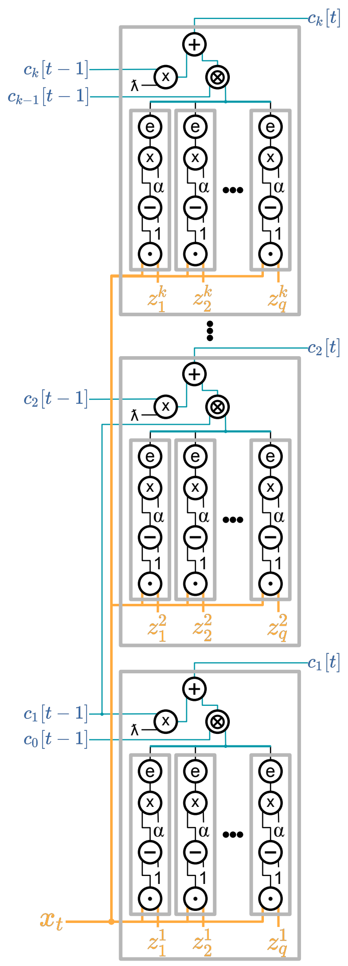


Figure V.1: The arithmetic circuit showing the computation of a single time point in RKN. Black lines represents a scalar value. Orange lines are d-dimensional vectors and blue lines depict q-dimensional vectors.

Corruption of a proxy: In case of a corruption of P_1 by a semi-honest adversary, there is no gain for the adversary, since it only receives the share of the masked eigenvectors and the masks utilized by the helper party to mask the eigenvalues. At the end, P_1 gets the share of the reciprocal of the square root of the eigenvalues as a secret shared result of MUL. Therefore, INVSQRT is secure against an adversary corrupting P_1. On the other hand, if an adversary corrupts P_0, then it can reduce the possible eigenvalues of the Gram matrix into a specific set considering that $\mathscr{M}$, τ, s, Δ and α are from a specific range. Even in the worst case that the adversary deduces all the eigenvalues, which is extremely unlikely and negligible, it cannot reconstruct and learn the Gram matrix since P_0 receives only the share of the eigenvectors, which is indistinguishable from a random matrix for P_0. Thus, INVSQRT is also secure against an adversary corrupting P_0.

Corruption of the helper: Since the values in $\mathscr{M}$, τ and s are from a specific range, the helper party can utilize the linear systems $\Lambda^{'} = \tau\Lambda + s$ and $Q' = MQ$ to narrow down the possible values of Q and Λ, and G implicitly. Unless a masked value in the linear systems is at one of its edge cases,

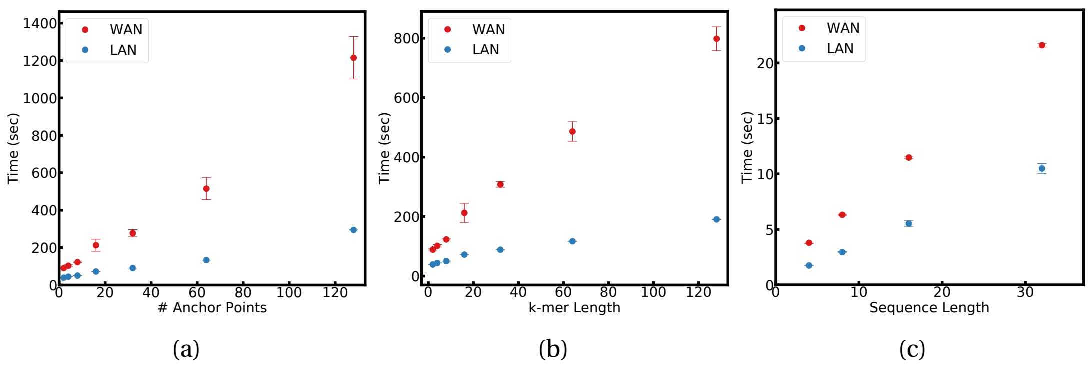


Figure V.2: The result of the experiments to analyze the execution time of CECILIA on both WAN and LAN settings for **(a)** varying number of anchor points for a fixed k-mer length and sequence length, **(b)** varying length of k-mers for a fixed number of anchor points and sequence length, and **(c)** varying length of sequences for a fixed number of anchor points and length of k-mer.

it is, however, not possible for the adversary to infer the actual value. It is plausible to assume that not every value is at the edge case in these linear systems. Furthermore, considering that obtaining the eigenvalues of G has a relatively high chance compared to obtaining the eigenvectors of G, the adversary cannot form a linear system based on the inferred eigenvalues resulting in a unique solution for the eigenvectors, eventually a Gram matrix. Considering all these infeasibilities of obtaining the eigenvalues and eigenvectors in addition to the computational infeasibility and lack of validation method of a brute-force approach, the adversary cannot obtain G by utilizing Λ' and Q'. Therefore, INVSQRT is secure against a semi-honest adversary corrupting the helper party with a high probability. □

Lemma 11. In case of the utilization of the outsourced inverse square root of the Gram matrices of the anchor points, the protocol ppRKN securely realizes the inference of a test sample via the pre-trained RKN model. If INVSQRT is utilized, then the protocol ppRKN privately performs the inference of a test sample via the pre-trained RKN model with a high probability.

Proof. In addition to the experimental proof of the correctness of ppRKN, based on the hybrid model, we can state that ppRKN correctly performs the private inference since it utilizes the methods of CECILIA which are already shown to perform their corresponding tasks correctly.

Corruption of a proxy by a semi-honest adversary: In case of utilization of the outsourced inverse square root of the Gram matrices, the private inference of RKN via ppRKN involves a series of MUL, ADD, EXP, MSB, MUX and CMP operations on secret shared data, and addition and multiplication between secret shared data and a public scalar. Since we proved or gave the references to the proof of the security of those methods, we can conclude that ppRKN is secure via the hybrid model. Instead of using the outsourced inverse square root of the Gram matrices, if we employ INVSQRT to compute those matrices, the adversary corrupting P_0 can narrow down the set of possible eigenvalues of the matrices to a relatively small set of values. Even in the worst case, the adversary cannot obtain the Gram matrix since it has only the share of the eigenvectors of the Gram matrix. For an adversary compromising P_1, there is no information gain at all. Therefore, ppRKN performs the fully private inference in case of the utilization of the outsourced inverse square root of the Gram matrices and

Protocol	Round Complexity
ADD	0
MUL	2
CMP	2
MUX	2
MSB	6
DP	2
EXP	26
INVSQRT	15

Table V.1: Communication round complexity of CECILIA

realizes the private inference with a high probability when INVSQRT is used.

Corruption of the helper party by a semi-honest adversary: In case of utilization of the outsourced inverse square root of the Gram matrices, considering the same reason we stated in the scenario where the adversary corrupts a proxy, we can conclude that ppRKN is secure via the hybrid model. Instead of using the outsourced inverse square root of the Gram matrices, if we employ INVSQRT to compute those matrices, the adversary corrupting the helper party can narrow down the set of possible values of the eigenvalues and the eigenvectors of the Gram matrix to a relatively small set of values. As we discuss in the security analysis of INVSQRT, such a reduction on the set is not enough for the adversary to reconstruct the Gram matrix with a high probability. Therefore, ppRKN is either fully secure or secure with a high probability depending on whether INVSQRT is used or not, respectively. □

V.6 Complexity Analysis of the Framework

In this section, we analyze the communication round complexity of the methods in CECILIA. Table V.1 summarizes the round complexities for each method in the framework. Note that we also give the analysis of the methods that we adapted from the other studies to give the comprehensive view of CECILIA. Except from ADD, the rest of the methods require communication between the parties. Most of the methods, namely MUL, CMP, MUX and DP, cost 2 communication rounds between the parties. MSB, on the other hand, requires 6 communication rounds between parties to obtain the most significant bit of a given secret shared value. One of the novel methods of CECILIA, which is INVSQRT, costs 15 communication rounds to compute the inverse square root of a secret shared Gram matrix. The other novel method that we proposed is EXP and the communication round complexity of this method is 26.

V.7 Results

V.7.1 Dataset

To replicate the experiments of the RKN and verify the correctness of ppRKN, we utilized the same dataset as Chen et al. [17], namely Structural Classification of Proteins (SCOP) version 1.67 [130] consisting of 85 fold recognition tasks, each of which contains protein sequences of different lengths, labeled as positive or negative.

We also created two synthetic test sets. One of them contains 5 sequences of length 128. We used this test set to evaluate the effects of the hyperparameters q and k on the execution time of CECILIA. The second test set consists of 5 sequences of different lengths to evaluate the effect of sequence length on execution time.

V.7.2 Experimental Setup

To conduct experiments, we used Amazon EC2 t2.xlarge instances. For the LAN setting, we selected all instances from the Frankfurt region, and for the WAN setting, we selected additional instances from London and Paris. We represent numbers with 64 bits whose least significant 20 bits are allocated to the fractional part. When computing the final mapping of the sequence, we chose INVSQRT to compute the inverse square root of the Gram matrices to provide the worst case runtime results, since the runtime of the experiments in which we directly use the outsourced inverse square root of the Gram matrices is significantly shorter than the runtime of the experiments in which we use INVSQRT to compute these matrices.

V.7.3 Correctness Analysis

We conducted the experiments to analyze the correctness of ppRKN on the LAN. We performed the predictions of test samples of a task of SCOP to verify that we can get the same predictions score as with RKN in plaintext. We selected the first task and trained a different RKN model for each combination of the parameters $q \in \{16, 32, 64, 128\}$ and $k \in \{5, 7, 10\}$. Afterwards, we split the parameters of the models into two arithmetic secret shares and sent the set of shares to the corresponding proxies. In addition to the parameters, we randomly selected 5 sequences from the test samples utilized by the RKN in this task and input these sequences into ppRKN by splitting the one-hot encoded versions of them into two arithmetic secret shares and sending them to the corresponding proxies. They privately performed the prediction of these test samples on the secret shared model. At the end of this process, we received the shares of the predictions from the proxies and performed the reconstruction to obtain these predictions in plaintext. When we compared the predictions of ppRKN with the predictions of RKN for these randomly selected test samples, the largest absolute difference between the corresponding predictions is less than 2×10^{-5}. Such close predictions of ppRKN suggest that CECILIA can correctly perform almost the same predictions as RKN without sacrificing the privacy of the test samples or of the model.

There are two reasons for the small deviation in prediction results. The first is the truncation operation at the end of the multiplication to preserve the format of the resulting number [34]. Although this causes only a small error for a single multiplication, it can add up for thousands of multiplications. Second, the utilized number format has inherently limited precision and this could cause a small error between the representation of the value in our format and the actual value itself. Such an error could also accumulate during the calculation and lead to a relatively high error in the end. Depending on the problem, one can address these problems by setting the number of bits for the fractional part to a higher number to increase the precision and minimize the overall error.

V.7.4 Execution Time Analysis

In addition to the verification of the correctness of ppRKN, we also examined the effects of the parameters of the RKN, namely the number of anchor points q, the length of the k-mers k and the length of the sequence s, on the execution time of ppRKN on both LAN and WAN. For this purpose, we used two synthetic datasets. To demonstrate the scalability of ppRKN to $q \in \{2,4,8,16,32,64,128\}$, we set $k = 16$ and $s = 128$. The results demonstrated that the runtime of CECILIA on both WAN and LAN settings scales close to linear to varying number of anchor points. Similarly, we evaluated the effect of $k \in \{2,4,8,16,32,64,128\}$ for a fixed $q = 16$ and $s = 128$. In this case, the execution time of ppRKN is best characterized with a linear line indicating the linear scalability of ppRKN to the length of the k-mer. We also experimented with $s \in \{4,8,16,32\}$ for a fixed $q = 4$ and $k = 4$ to analyze the impact of s on execution time. Similar to k, ppRKN scales linearly to s. Figure V.2 summarizes these results and demonstrates the trends in the execution time of ppRKN for different parameters.

V.8 Conclusion

In this study, we propose a new comprehensive secure framework, called CECILIA, based on 3-party computation to enable privacy preserving computation of complex functions utilized in machine learning algorithms, some of which have not been addressed before such as the exact exponential computation and the inverse square root of a Gram matrix. To the best of our knowledge, CECILIA enables privacy preserving inference of RKNs for the first time. We demonstrate the correctness of CECILIA on the LAN using the tasks of the original RKN paper. The experiments demonstrate that CECILIA can give almost the exact same prediction result as one could obtain from the RKN without privacy. This proves the correctness of the proposed methods. Moreover, we evaluate the performance of CECILIA in terms of execution time on two synthetic datasets. We carry out these analyses on WAN and show that CECILIA scales linearly with the length of the k-mers and the length of the input sequence, and almost linearly with the number of anchor points. It is important to note that CECILIA is not just limited to the privacy preserving inference on RKN but also capable of realizing other machine learning algorithms employing the provided building blocks. In future work, building on CECILIA, we plan to address the privacy preserving training of the RKN and other state-of-the-art machine learning algorithms.

V.9 Supplement

V.9.1 Number Format

To illustrate the number format, let n be the number of bits to represent numbers, f be the number of bits allocated for the fractional part and $\mathbb{S}$ be the set of values that can be represented in this number format, one can convert $x \in \mathbb{R}$ to $\hat{x} \in \mathbb{S}$ as follows:

$$\hat{x} = \begin{cases} \lfloor x * 2^f \rfloor & x \geq 0 \\ 2^n - \lfloor |x * 2^f| \rfloor & x < 0 \end{cases} \tag{V.4}$$

For example, $x = 3.42$ is represented as $\hat{x} = 112066$, which, if you omit the leading zeros, is 11011010111000001010001 in binary. The two most significant bits, the 21st and 22nd bits, are used to represent 3 and the remainder represents 0.42.

The choice of f depends on the task for which CECILIA is used. If the numbers in the task are mostly small or they need to be as accurate as possible, the precision of the representation of the numbers is crucial. In such a case, a higher value for f is required to allow for more decimal places of the original value, which in turn requires sacrifices in the upper limit of the exponential to be calculated. However, if the numbers that appear during the process are large, f must be set to lower values to allow the integer part to represent larger numbers with fewer decimal places.

In our experiments with CECILIA, we represent the numbers with 64 bits and set f to 20 to have a relatively high precision in representing the numbers and a sufficient range for the exponential. Although we have used the 64 bit representation in the rest of the explanations of the functions in CECILIA, CECILIA can also work with a different number of bit representations. The reason for our choice is to take advantage of the natural modular operation of most modern CPUs, which operate with 64 bits. In this way, we avoid performing modular operations after each arithmetic operation performed on the shares. However, one can still use a larger or smaller number of bits on 64 bit CPUs by performing the modular operations manually.

V.9.2 Security Analysis of EXP

In this part, we give the security analysis of EXP.

Lemma 12. The protocol EXP securely computes the exponential of a publicly known base raised to the power of a secret shared value.

Proof. We begin the proof by showing the correctness of the method. Let x be the power whose representation in our number format is $\langle x \rangle$ and b be the publicly known base. One of the proxies computes $C_p = \{\dots, b^8, b^4, b^2, b, b^{1/2}, b^{1/4}, b^{1/8}, \dots\}$ and $C_n = \{\dots, b^{-8}, b^{-4}, b^{-2}, b^{-1}, b^{-1/2}, b^{-1/4}, b^{-1/8}, \dots\}$ and the other generates a corresponding set of 0s for C_p and C_n. These values in C_p and C_n correspond to b^{2^i} and $b^{-1 \cdot 2^i}$, respectively, for $i \in \{(n-f), \dots, 2, 1, 0, -1, -2, \dots, -f\}$ assuming that the corresponding bit value is 1. They choose one of these sets based on the sign of x and let C be the selected set. Afterwards, they must choose between $c_j \in C$ and 1 depending on $\langle x \rangle_j$ where $j \in \{0, 1, \dots, n\}$. For this selection, they repeat the most significant bit operation and bit shifting of x to the left by 1 until they have processed each bit. Once they have the correct set of contributions, they basically multiply all of those contributions to obtain the result of the exponential. This proves the correctness of EXP.

Corruption of a proxy: At the beginning, since the adversary corrupting a proxy knows only one share of the power x, that is either x_0 or x_1, it cannot infer any information about the other share. The first step of the exponential is to compute the possible contribution of every bit of positive and negative power. This is publicly known. The following step is to select between these contribution depending on the result of MSB(x) by using MUX. Since both MSB and MUX are secure, the adversary can neither infer anything about x_{1-j} nor relate the share of the result it obtains to x in general. In the next step, they obtain each bit of x in secret shared form by using MSB and bit shifting on the shares of x. Considering the proven security of MSB and the shifting being simply multiplication of each share by 2, there is no information that the adversary could obtain. Afterwards, the proxies select the correct contributions by employing MUX. Since MUX gives the fresh share of what is selected, the adversary cannot associate the inputs to the output. The last step is to multiply these selected contributions via MUL, which is also proven to be secure. Therefore, we can conclude that EXP is secure against a semi-honest adversary corrupting a proxy.

Corruption of the helper: Since the task of the helper party in the computation of the exponential of a secret shared power is either to provide multiplication triples or to perform the required computation on the masked data, there is nothing that the adversary corrupting the helper party could learn about x. Therefore, it is fair to state that EXP is secure against a semi-honest adversary corrupting the helper. $\square$

VI ppAURORA: Privacy Preserving Area Under Receiver Operating Characteristic and Precision-Recall Curves with Secure 3-Party Computation

Ali Burak Ünal **Mete Akgün** **Nico Pfeifer**

Abstract

Computing an 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 machine learning and methods based on cryptographic approaches if the datasets cannot be shared or evaluated jointly at one place. In this setting, it can also be a problem to compute the global performance measure like an AUC, since the labels might also contain privacy-sensitive information. There have been approaches based on ϵ-differential privacy to deal with 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 framework, called ppAURORA, with private merging of sorted lists and novel methods for comparing two secret-shared values, selecting between two secret-shared values, converting the modulus, and performing division to compute the exact AUC as one could obtain on the pooled original test samples. With ppAURORA computation of the exact area under precision-recall curve and receiver operating characteristic curve is even possible when ties between prediction confidence values exist. To show the applicability of ppAURORA, we use it to evaluate a model trained to predict acute myeloid leukemia therapy response and we also assess its scalability via experiments on synthetic data. The experiments show that we efficiently compute exactly the same AUC with both evaluation metrics in a privacy preserving manner as one can obtain on the pooled test samples in the plaintext domain. Our solution provides security against semi-honest corruption of at most one of the servers performing the secure computation.

VI.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 [34, 2, 131, 132, 86, 133, 134, 135]. 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. Even though there are no studies in the literature enabling such a computation for the precision-recall (PR) curve, there are several differential privacy based approaches in the literature for the receiver operating characteristic (ROC) curve [136, 137, 138]. Briefly, they protect the privacy of the data by introducing noise into the computation so that one cannot obtain the original data employed in the computation. However, due to the nature of differential privacy, the resulting AUC is different from the one which could be obtained by using non-perturbed prediction confidence values (PCVs). For private computation of the exact AUC, there exists no approach in the literature to the best of our knowledge.

In this paper, we propose the **p**rivacy **p**reserving **a**rea **u**nder **r**eceiver **o**perating characteristic and precision-**r**ec**a**ll curves (ppAURORA) based on a secure 3-party computation framework to address the necessity of an efficient, private and secure computation of the exact AUC. We compute the area under the PR curve (AUPR) and ROC curve (AUROC) with ppAURORA. 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 by using PCVs with no tie. In case of a tie of PCVs of samples from different classes, it just approximates the metric based on the order of the samples, having a problem when values of both axes change at the same time. In order to compute the exact AUC even in case of a tie, we introduce the second version with a slightly higher communication cost than the first approach. Along with the AUC, both are capable of protecting the information of the number of samples belonging to the classes from all participants of the computation, which could be used to obtain the order of the labels of the PCVs [139]. 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 [140]. We utilized 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.

We introduce a novel 3-party computation framework to achieve privacy preserving AUC computation with ppAURORA. The framework consists of privacy preserving sorting and four operations with novel and efficient versions, which are select share (MUX), modulus conversion (MC), compare (CMP) and division (DIV). MUX is designed to select one of two secret shares based on a secret shared bit value. MC privately converts the ring of a secret shared value from $2^{\ell-1}$ to 2^{ℓ}. CMP compares two secret shared values, determines if the first argument is larger than the second one without revealing values and splits the result in a secret shared way. DIV performs the division of two secret shared values without sacrificing the privacy of the values. Note that our new DIV is specifically customized for efficient and secure AUC computation.

VI.2 Scenarios

In this section, we describe the scenarios at which ppAURORA is applicable. Note that it is not limited to these scenarios.

End-to-end MPC-based Collaborative Learning: Recently, researchers proposed multi-party computation (MPC) based training and testing of several machine learning algorithms [34, 2, 133]. Their approaches can train the model privately and collaboratively, and make predictions on the test data of the data sources involved in the computation. However, the privacy preserving collaborative evaluation of the model is lacking. To fulfill such a gap, one can integrate ppAURORA at the end of the process once the PCVs are secret shared among two computing parties. One possible scenario would be that one wants to build a model for predicting hospitalization times for COVID-19 patients. Usually, the personal data cannot be shared easily, due to the private nature of the data, but even sharing the hospitalization times might be problematic in case the hospitals do not want to allow competitors to learn about this piece of information. Nevertheless, one can assume that a model built on data from many hospitals will perform much better than models built on individual datasets and that the global AUC will allow for a better model selection than an average of locally computed AUCs. This privacy preserving global AUC computation can be performed with our ppAURORA.

Evaluation of Models Trained by Federated Learning: In some cases, the data is not allowed to be shared at all. For such cases, federated learning has been widely utilized to train a collaborative machine learning model without gathering the data in one place. Each data source updates the model by its own data in an online learning process and passes the model to the next data source until the model converges or the iteration limit is reached. Once the data sources obtain the trained model, they make the predictions of their own test data. In order to collaboratively evaluate the performance of the model, they can utilize ppAURORA.

VI.3 Preliminaries

Security Model: In this study, 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 [141, 142] 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 [142] 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 by Araki et al. [143]. The privacy is formalized by saying that a malicious party, which arbitrarily deviates 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 three different rings $\mathbb{Z}_L$, $\mathbb{Z}_K$ and $\mathbb{Z}_P$ where $L = 2^\ell$, $K = 2^{\ell-1}$, $P = 67$ and $\ell = 64$. We denote two shares of x over $\mathbb{Z}_L$, $\mathbb{Z}_K$ and $\mathbb{Z}_P$ with $(\langle x\rangle_0, \langle x\rangle_1)$, $(\langle x\rangle_0^K, \langle x\rangle_1^K)$ and $(\langle x\rangle_0^P, \langle x\rangle_1^P)$, respectively. If a value x is shared over the ring $\mathbb{Z}_P$, each bit of x is additively shared in $\mathbb{Z}_P$. This means x is shared as a vector of 64 shares where each share takes a value between 0 and 66. We also use boolean sharing of a single bit which is denoted with $(\langle x\rangle_0^B, \langle x\rangle_1^B)$.

VI.3.1 Secure Multi-party Computation

Secure multi-party computation was proposed in the 1980s [25, 26]. These studies showed that multiple parties can compute any function on inputs without learning anything about the inputs of the other parties. Let us assume that there are n parties $I_1, \cdots, I_n$ and I_i has a private input x_i for $i \in \{1, \ldots, n\}$. All parties want to compute the arbitrary function $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ and get the result y_i. MPC allows the parties to compute the function through an interactive protocol and I_i to learn only y_i.

We first explain the 2-out-of-2 additive secret sharing and how addition (ADD) and multiplication (MUL) are computed. In additive secret sharing, an ℓ-bit value x is shared additively in the ring $\mathbb{Z}_L$ 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 I_i knows only $\langle x\rangle_i$ and $i \in \{0, 1\}$. All arithmetic operations are performed in the ring $\mathbb{Z}_L$. For additive secret sharing, we use protocols based on Beaver's multiplication triples [105].

Addition: $\langle z \rangle = \langle x \rangle + \langle y \rangle$. I_i locally computes $\langle z \rangle_i = \langle x \rangle_i + \langle y \rangle_i$. In order to compute the addition of a shared value $\langle x \rangle$ and a constant c, I_i locally computes $\langle z \rangle_i = \langle x \rangle_i + c$ and I_{1-i} locally computes $\langle z \rangle_{1-i} = \langle x \rangle_{1-i}$ for i is either 0 or 1.

Multiplication: $\langle z \rangle = \langle x \rangle \cdot \langle y \rangle$. Multiplication is performed using a pre-computed multiplication triple $\langle c \rangle_i = \langle a \rangle_i \cdot \langle b \rangle_i$ [105]. I_i computes $\langle e \rangle_i = \langle x \rangle_i - \langle a \rangle_i$ and $\langle f \rangle_i = \langle y \rangle_i - \langle b \rangle_i$. I_i sends $\langle e \rangle_i$ and $\langle f \rangle_i$ to I_{1-i}. I_i reconstructs e and f, and then computes $\langle z \rangle_i = i \cdot e \cdot f + f \cdot \langle a \rangle_i + e \cdot \langle b \rangle_i + \langle c \rangle_i$. The computation of the multiplication triple is performed via homomorphic encryption or oblivious transfer. I_i cannot perform multiplication locally.

VI.3.2 Area Under Curve

One of the most common ways to summarize the plot-based model evaluation metrics is area under curve (AUC) which measures the area under the curve. It is applicable to various different evaluation metrics. In this study, we employ AUC to measure the area under the ROC curve and the PR curve.

Area Under ROC Curve (AUROC): In machine learning problems with binary outcome, the ROC curve is very effective to take the sensitivity and the specificity of the 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 ROC curve (AUROC). Let us assume that M is the number of test samples, $V \in [0,1]^M$ contains 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 that there is no tie in V, the privacy-friendly AUROC computation is as follows:

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

This formula just approximates the exact AUROC in case of a tie in V depending on the order of the samples. As an extreme example, let V have 10 samples with the same PCV. The first 5 samples have label 1 and the second 5 samples have label 0. Such a setting outputs $AUROC = 1$. As a contrary, if we have samples with 0 at the beginning and samples with 1 later, we obtain $AUROC = 0$. In order 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 are different for $0 \le |\xi| \le M$ where $|\xi|$ denotes the size of the vector. Assuming that $\xi[0] = 0$, the computation of AUROC can be done as follows:

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

As Equation VI.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.

Area Under PR Curve (AUPR): PR curve evaluates the models with binary outcome by plotting recall on the x-axis and precision on the y-axis, and it is generally preferred over AUROC for scenarios

SecureNN [2]			Our Framework		
Protocol	Rounds	Communication	Protocol	Rounds	Communication
SelectShare	2	5ℓ	MUX	2	6ℓ
ShareConvert	4	$4\ell\log P + 6\ell$	ModulusConversion	3	$4\ell\log P + 6\ell$
ComputeMSB	5	$4\ell\log P + 13\ell$	Compare	5	$4\ell\log P + 11\ell$
DIV	$10\ell_D$	$(8\ell\log P + 24\ell)\ell_D$	DIV	2	6ℓ

Table VI.1: Complexity comparison of distinguishing protocols of ppAURORA with SecureNN [2]

with class imbalances. The AUC summarizes this plot by measuring the area under the PR curve (AUPR). Since both precision and recall can change at the same time even without a tie, we measure the area by using the Equation VI.2 where T being the precision and F being the recall.

VI.4 Framework

In this section, we give the definitions of basic operations of the framework that we use in ppAURORA. Note that we include only the operations with novelty due to the page limit. One can refer to the Supplement for the other operations.

Selecting One of Two Secret Shared Values: Algorithm VI.1 performs 3-party computation to select one of two secret shared values based on the secret shared bit value (functionality $\mathcal{F}_{\mathsf{MUX}}$). At the beginning of the protocol, S_0 and S_1 hold $(\langle x\rangle_0, \langle y\rangle_0, \langle b\rangle_0)$ and $(\langle x\rangle_1, \langle y\rangle_1, \langle b\rangle_1)$, respectively. At the end of the secure computation, S_0 and S_1 hold the fresh shares of $z = x - b(x-y)$. We utilize the randomized encoding of multiplication [144]. As shown in Equation VI.3, we need to multiply two values owned by different parties in the computation of $\langle b\rangle_0(\langle x\rangle_1 - \langle y\rangle_1)$ and $\langle b\rangle_1(\langle x\rangle_0 - \langle y\rangle_0$. We assume that S_2 is the computation party and performs these multiplications via the randomized encoding.

$$\begin{aligned} z &= x - b(x-y) \\ &= \langle x\rangle_0 + \langle x\rangle_1 - \langle b\rangle_0(\langle x\rangle_0 - \langle y\rangle_0) - \langle b\rangle_1(\langle x\rangle_1 - \langle y\rangle_1) \\ &\quad - \langle b\rangle_0(\langle x\rangle_1 - \langle y\rangle_1) - \langle b\rangle_1(\langle x\rangle_0 - \langle y\rangle_0) \end{aligned} \tag{VI.3}$$

Modulus Conversion: Algorithm VI.2 describes our 3-party protocol realizing the functionality $\mathcal{F}_{\mathsf{MC}}$ that converts shares over $\mathbb{Z}_K$ to fresh shares over $\mathbb{Z}_L$ where $L = 2K$. Assuming that S_0 and S_1 have the shares $\langle x\rangle_0^K$ and $\langle x\rangle_1^K$, respectively, the first step for S_0 and S_1 is to mask their shares by using the shares of the random value $r \in \mathbb{Z}_K$ sent by S_2. Afterwards, they reconstruct $(x + r) \in \mathbb{Z}_K$, by first computing $\langle y\rangle_i^K = \langle x\rangle_i^K + \langle r\rangle_i^K$ for $i \in \{0,1\}$ and sending these values to each other. Along with the shares of $r \in \mathbb{Z}_K$, S_2 also sends the information in boolean shares telling whether the summation of the shares of r wraps so that S_0 and S_1 can convert r from the ring $\mathbb{Z}_K$ to the ring $\mathbb{Z}_L$. Once they reconstruct $y \in \mathbb{Z}_K$, S_0 and S_1 can change the ring of y to $\mathbb{Z}_L$ by adding K to one of the shares of y if $\langle y\rangle_0^K + \langle y\rangle_1^K$ wraps. After conversion, the important detail regarding $y \in \mathbb{Z}_L$ is to fix the value of it. In case $(x + r) \in \mathbb{Z}_K$ wraps, which we identify by using PC, depending on the boolean share of the outcome of PC, S_0 or S_1 or both add K to their shares. If both adds, this means that there is no addition to the value of $y \in \mathbb{Z}_L$. At the end, S_i subtracts $r_i \in \mathbb{Z}_L$ from $y_i \in \mathbb{Z}_L$ and obtains $x_i \in \mathbb{Z}_L$ for $i \in \{0,1\}$.

Comparison of Two Secret Shared Values: Algorithm VI.3 gives the definition of the 3-party

```
1 Algorithm MUX()
      input  : $S_0$ and $S_1$ hold $(\langle x\rangle_0, \langle y\rangle_0, \langle b\rangle_0)$ and $(\langle x\rangle_1, \langle y\rangle_1, \langle b\rangle_1)$, respectively.
      output : $S_0$ and $S_1$ get $\langle z\rangle_0$ and $\langle z\rangle_1$, respectively, where $z = x - b(x-y)$.
2     $S_0$ and $S_1$ hold four common random values $r_i$ where $i \in \{0,1,2,3\}$
3     $S_0$ computes $M_1 = \langle x\rangle_0 - \langle b\rangle_0(\langle x\rangle_0 - \langle y\rangle_0) + r_1\langle b\rangle_0 + r_2(\langle x\rangle_0 - \langle y\rangle_0) + r_2 r_3$, $M_2 = \langle b\rangle_0 + r_0$,
        $M_3 = \langle x\rangle_0 - \langle y\rangle_0 + r_3$
4     $S_0$ sends $M_2$ and $M_3$ to $S_2$
5     $S_1$ computes $M_4 = \langle x\rangle_1 - \langle b\rangle_1(\langle x\rangle_1 - \langle y\rangle_1) + r_0(\langle x\rangle_1 - \langle y\rangle_1) + r_0 r_1 + r_3\langle b\rangle_1$,
        $M_5 = (\langle x\rangle_1 - \langle y\rangle_1) + r_1$, $M_6 = \langle b\rangle_1 + r_2$
6     $S_1$ sends $M_5$ and $M_6$ to $S_2$
7     $S_2$ computes $M_2 M_5 + M_3 M_6 = z$
8     $S_2$ divides $z$ into two shares $(\langle z\rangle_0 + \langle z\rangle_1)$ and sends $\langle z\rangle_0$ and $\langle z\rangle_1$ to $S_0$ and $S_1$, respectively
9     $S_0$ computes $\langle z\rangle_0 = M_1 - \langle z\rangle_0$
10    $S_1$ computes $\langle z\rangle_1 = M_4 - \langle z\rangle_1$
```

Algorithm VI.1: Select Share (MUX)

protocol for the functionality $\mathcal{F}_{\mathsf{CMP}}$ comparing two secret shared values x and y, and outputs zero if $x \geq y$, 1 otherwise. In Algorithm VI.3, we find the value of $(x-y) \wedge 2^{L-1}$ indicating the most significant bit (MSB) of $(x-y)$. First, S_i where $i \in \{0,1\}$ computes $\langle d\rangle_i^K = (\langle x\rangle_i^L - \langle y\rangle_i^L) \mod K$. S_i converts the ring of d from $\mathbb{Z}_K$ to $\mathbb{Z}_L$ by calling MC. Finally, S_0 and S_1 subtract the shares of d over $\mathbb{Z}_L$ from the shares of $x-y$ and map the result to 1 if it equals K, otherwise 0.

Division: Algorithm VI.4 gives the definition of the 3-party protocol realizing the functionality $\mathcal{F}_{\mathsf{DIV}}$ that computes x/y. S_0 and S_1 hold shares of x and y, and DIV outputs fresh shares of x/y. DIV computes x/y correctly when S_0 and S_1 know the upper bound for x and y. Thus, it is not a general purpose method over $\mathbb{Z}_L$. We rather specifically designed it for ppAURORA since S_0 and S_1 know the upper bound of inputs to DIV in the computation of the AUC.

VI.4.1 Complexities of Our Protocols

There are some frameworks in the literature that also perform 3-party secure computation. The recently published SecureNN [2] has shown successful results in terms of performance in private CNN training. We compared our secure protocols with the corresponding building blocks from SecureNN that one needs to use for computing AUC. Table VI.1 summarizes the comparison. It is clear that our protocols have lower costs. Therefore, we can deduce that our framework outperforms SecureNN in AUC computation. Our novel protocols (MUX, MC, CMP, DIV) require at most 5 communication rounds, which is the most important factor affecting the performance of secure computing protocols. Our new protocols show an improvement in terms of round complexities.

VI.5 ppAURORA Computation

In this section, we give the description of our protocol for ppAURORA computation. In ppAURORA, 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 AUC. The protocol starts with outsourcing by the data sources. Afterward, the servers perform the desired calculation privately. Finally, they send the shares of the result back to the data sources. The

```
1  Algorithm MC()
       input  : $S_0$ and $S_1$ hold $\langle x\rangle_0^K$ and $\langle x\rangle_1^K$, respectively
       output : $S_0$ and $S_1$ get $\langle x\rangle_0$ and $\langle x\rangle_1$, respectively
2      $S_0$ and $S_1$ hold a common random bit $n'$
3      $S_2$ picks a random numbers $r \in \mathbb{Z}_{2^{L-1}}$ and generates $\langle r\rangle_0^K$, $\langle r\rangle_1^K$, $\{\langle r[j]\rangle_0^p\}_{j\in[\ell]}$ and $\{\langle r[j]\rangle_1^p\}_{j\in[\ell]}$.
4      $S_2$ computes $w = \text{isWrap}(\langle r\rangle_0^K, \langle r\rangle_1^K, K)$ and divides $w$ into two boolean shares $w_0^B$ and $w_1^B$
5      $S_2$ sends $\langle r\rangle_i^K$, $\{\langle r[j]\rangle_i^p\}_{j\in[\ell]}$ and $w_i^B$ to $S_i$, for each $i \in \{0,1\}$
6      For each $i \in \{0,1\}$, $S_i$ executes Steps 7-8
7      $\langle y\rangle_i^K = \langle x\rangle_i^K + \langle r\rangle_i^K$
8      $S_i$ reconstructs $y$ by exchanging shares with $S_{1-i}$
9      $n_i^B = \mathsf{PC}(\{\langle r[j]\rangle_i^p\}_{j\in[\ell]}, y, n')$
10     $S_0$ computes $n_i^B = n_i^B \oplus n'$
11     For each $i \in \{0,1\}$, $S_i$ computes $c_i^B = w_i^B \oplus n_i^B$
12     $S_0$ computes $\langle y\rangle_0 = \langle y\rangle_0^K + \text{isWrap}(\langle y\rangle_0^K, \langle y\rangle_1^K, K)\cdot K$
13     $S_1$ sets $\langle y\rangle_1 = \langle y\rangle_1^K$
14     For each $i \in \{0,1\}$, $S_i$ computes $\langle x\rangle_i = \langle y\rangle_i - (\langle r\rangle_i^K + c_i^B \cdot K)$
```

Algorithm VI.2: Modulus Conversion (MC)

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 T_i according to PCVs in descending order and divides it into two additive shares T_{i_0} and T_{i_1}, and sends T_{i_0} and T_{i_1} to S_0 and S_1, respectively. We refer to S_0 and S_1 as *proxies*.

Sorting: After the outsourcing phase, S_0 and S_1 obtain the shares of individually sorted lists of PCVs of the data owners. Afterwards, the proxies need to perform a merging operation on each pair of individually sorted lists and continue with the merged lists 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 that determines the number of PCVs that will be added to the global sorted list in each iteration for $a \in \mathbb{N}$, T_{i_k} and T_{j_k} be the shares of two individually sorted lists of PCVs in S_ks for $k \in \{0,1\}$ and $|T_i| \geq |T_j|$ where $|.|$ is size operator. At the beginning, the proxies privately compare the lists elementwise. They utilize the results of the comparison in MUXs to privately exchange the shares of PCVs in each pair if the PCV in T_j is larger than the PCV in T_i. In the first MUX, they input the share in T_{i_k} to MUX first and then the share in T_{j_k} along with the share of the result of the comparison to select the larger of the PCVs and move it to T_{i_k}. In the second MUX, they reverse the order to select the smaller of the PCVs and move it to T_{j_k}. We call this stage *shuffling*. Then, they move the top PCV of T_{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. Until finishing up the PCVs in T_{i_k}, the proxies follows shuffling-moving cycle.

The purpose of the shuffling is to 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

```
1 Algorithm CMP()
      input  : $S_0$ and $S_1$ hold $(\langle x\rangle_0, \langle y\rangle_0)$ and $(\langle x\rangle_1, \langle y\rangle_1)$, respectively
      output : $S_0$ and $S_1$ get $\langle z\rangle_0$ and $\langle z\rangle_1$, respectively, where $z$ is equal to zero if $x >= y$ and 1
               otherwise
2     $S_0$ and $S_1$ hold a common random bit $f$
3     For each $i \in \{0,1\}$, $S_i$ executes Steps 4-9.
4     $\langle d\rangle_i^K = (\langle x\rangle_i - \langle y\rangle_i) \mod K$
5     $\langle d\rangle_i = \mathsf{MC}(\langle d\rangle_i^K)$.
6     $\langle z\rangle_i = \langle x\rangle_i - \langle y\rangle_i - \langle d\rangle_i$.
7     $\langle a[0]\rangle_i = ifK - \langle z\rangle_i$
8     $\langle a[1]\rangle_i = i(1-f)K - \langle z\rangle_i$
9     $S_i$ sends $\langle a\rangle_i$ to $S_2$
10    $S_2$ reconstructs $a[j]$ where $j \in \{0,1\}$ and computes $a[j] = a[j]/K$
11    $S_2$ creates two fresh shares of $a[j]$ where $j \in \{0,1\}$ sends them to $S_0$ and $S_1$
12    For each $i \in \{0,1\}$, $S_i$ executes Step 13
13    $\langle z\rangle_i = \langle a[f]\rangle_i$
```

Algorithm VI.3: Comparison of two secret shared values (CMP)

```
1 Algorithm DIV()
      input  : $S_0$ and $S_1$ hold $(\langle x\rangle_0, \langle y\rangle_0)$ and $(\langle x\rangle_1, \langle y\rangle_1)$, respectively.
      output : $S_0$ and $S_1$ get $\langle z\rangle_0$ and $\langle z\rangle_1$, respectively, where $z = x/y$.
2     $S_0$, $S_1$ and $S_2$ know the common scaling factor $F$
3     $S_0$ and $S_1$ know the upper limit of $x$ and $y$, which is denoted by $U$
4     $S_0$ and $S_1$ hold two common random values $r_0$ and $r_1$, where $r_0 < \lfloor L/2U \rfloor$ and $r_1 < \lfloor L/2U \rfloor$
5     For each $i \in \{0,1\}$, $S_i$ executes Steps 6-7.
6     $S_i$ computes $\langle a\rangle_i = r_1\langle x\rangle_i + r_0\langle y\rangle_i$ and $\langle b\rangle_i = r_1\langle y\rangle_i$
7     $S_i$ sends $\langle a\rangle_i$ and $\langle b\rangle_i$ to $S_2$
8     $S_2$ reconstructs $a$ and $b$ and computes $c = aF/b$
9     $S_2$ creates two shares of $c$ denoted by $\langle c\rangle_0$ and $\langle c\rangle_1$, and sends them to $S_0$ and $S_1$, respectively
10    $S_0$ computes $\langle z\rangle_0 = \langle c\rangle_0$
11    $S_1$ computes $\langle z\rangle_1 = \langle c\rangle_1 - r_0F/r_1$
```

Algorithm VI.4: Division (DIV)

list. The highest possible chance of a matching is 50%. This results in a very low chance of guessing the matching of whole PCVs in the list. Regarding the effect of δ on the privacy, it is important to note that δ needs to be an odd number to make sure that shuffling always leads to increment in the number of candidates. Even value of δ may cause ineffective shuffling during the sorting. Furthermore, $\delta = 1$ 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. However, the execution time of sorting with $\delta = 1$ 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 $|T_{j_k}| = 1$ 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 Supplement.

```
input  :$\langle T\rangle_i = (\{\langle con_1\rangle_i, \langle label_1\rangle_i\},\ldots,\{\langle con_M\rangle_i, \langle label_M\rangle_i\})$, $\langle T\rangle_i$ is a share of the global sorted
        list of PCVs, and labels
1  For each $i \in \{0,1\}$, $S_i$ executes Steps 2-11
2  $\langle TP\rangle_i \leftarrow 0$, $\langle P\rangle_i \leftarrow 0$, $\langle pFP\rangle_i \leftarrow 0$, $\langle N\rangle_i \leftarrow 0$
3  foreach item $\langle t\rangle \in \langle T\rangle$ do
4      $\langle TP\rangle_i \leftarrow \langle TP\rangle_i + \langle t.label\rangle_i$
5      $\langle P\rangle_i \leftarrow \langle P\rangle_i + i$
6      $\langle FP\rangle_i \leftarrow \langle P\rangle_i - \langle TP\rangle_i$
7      $\langle A\rangle_i \leftarrow \mathsf{MUL}(\langle TP\rangle_i, \langle FP\rangle_i - \langle pFP\rangle_i)$
8      $\langle N\rangle_i \leftarrow \langle N\rangle_i + \langle A\rangle_i$
9      $\langle pFP\rangle_i \leftarrow \langle FP\rangle_i$
10 $\langle D\rangle_i \leftarrow \mathsf{MUL}(\langle TP\rangle_i, \langle FP\rangle_i)$
11 $\langle ROC\rangle_i \leftarrow \mathsf{DIV}(\langle N\rangle_i, \langle D\rangle_i)$
```
Algorithm VI.5: Secure AUROC computation without ties

VI.5.1 Secure Computation of AUROC

Once S_0 and S_1 obtain the global sorted list of PCVs, they calculate the AUROC based on this list by employing one of the versions of AUROC depending on whether there exists tie in the list.

Secure AUROC Computation without Ties: In Algorithm VI.5, we compute the AUROC as shown in Equation VI.1 because we assume 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 where $AUROC = N/D$. S_i for $i \in \{0,1\}$ knows the number of test samples M. Thus S_i can determine the upper bound for N and D and DIV can be used to calculate $AUROC = N/D$. With the help of high numeric value precision of the results, most of the machine learning algorithms yield different PCVs for samples. Therefore, such an approach to compute the AUROC is applicable to most of the machine learning tasks. However, in case of a tie between samples from two classes in the PCVs, it does not guarantee that it gives the exact AUROC. Depending on the order of the samples, it approximates the score. To have a more accurate AUROC, we will propose another version with a slightly higher communication cost in the next section.

Secure AUROC Computation with Ties: To detect ties in the list of PCVs, S_0 and S_1 compute the difference between each PCV and its following PCV. S_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 S_3 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 test samples. Then, they send the list of shares to S_2. S_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 VI.7. In Algorithm VI.6, S_0 and S_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 the Equation VI.2. In Algorithm VI.6, DIV method is used because the upper limit for the numerator and denominator is known, as in the AUROC computation described in the previous paragraph.

Secure AUPR Computation: As in the AUROC computation described in the previous paragraph, S_0 and S_1 map a PCV in the global sorted list to zero if it equals to the previous PCV and one otherwise

```
    input  : $\langle T\rangle_i = (\{\langle con_1\rangle_i, \langle label_1\rangle_i\}, , ..., \{\langle con_M\rangle_i, \langle label_M\rangle_i\})$, $\langle T\rangle_i$ is a share of the global sorted
             list of PCVs, and labels
 1  For each $i \in \{0,1\}$, $S_i$ executes Steps 2-18
 2  $\langle TP\rangle_i \leftarrow 0$, $\langle P\rangle_i \leftarrow 0$, $\langle pFP\rangle_i \leftarrow 0$, $\langle pTP\rangle_i \leftarrow 0$, $\langle N_1\rangle_i \leftarrow 0$, $\langle N_2\rangle_i \leftarrow 0$
 3  foreach item $\langle t\rangle_i \in \langle T\rangle_i$ do
 4      $\langle TP\rangle_i \leftarrow \langle TP\rangle_i + \langle t.label\rangle_i$
 5      $\langle P\rangle_i \leftarrow \langle P\rangle_i + i$
 6      $\langle FP\rangle_i \leftarrow \langle P\rangle_i - \langle TP\rangle_i$
 7      $\langle A\rangle_i \leftarrow \mathsf{MUL}(\langle pTP\rangle_i, \langle FP\rangle_i - \langle pFP\rangle_i)$
 8      $\langle A\rangle_i \leftarrow \mathsf{MUL}(\langle A\rangle_i, \langle t.con\rangle_i)$
 9      $\langle N_1\rangle_i \leftarrow \langle N_1\rangle_i + \langle A\rangle_i$
10      $\langle A\rangle_i \leftarrow \mathsf{MUL}(\langle TP\rangle_i - \langle pTP\rangle_i, \langle FP\rangle_i - \langle pFP\rangle_i)$
11      $\langle A\rangle_i \leftarrow \mathsf{MUL}(\langle A\rangle_i, \langle t.con\rangle_i)$
12      $\langle N_2\rangle_i \leftarrow \langle N_2\rangle_i + \langle A\rangle_i$
13      $\langle pre\_FP\rangle_i \leftarrow \mathsf{MUX}(\langle pFP\rangle_i, \langle FP\rangle_i, \langle t.con\rangle_i)$
14      $\langle pre\_TP\rangle_i \leftarrow \mathsf{MUX}(\langle pTP\rangle_i, \langle TP\rangle_i, \langle t.con\rangle_i)$
15  $\langle N\rangle_i \leftarrow 2 \cdot \langle N_1\rangle_i + \langle N_2\rangle_i$
16  $\langle D\rangle_i \leftarrow 2 \cdot \mathsf{MUL}(\langle TP\rangle_i, \langle FP\rangle_i)$
17  $\langle ROC\rangle_i \leftarrow \mathsf{DIV}(\langle N\rangle_i, \langle D\rangle_i)$
```

Algorithm VI.6: Secure AUROC computation with tie

by running the Algorithm VI.7. Then, we use Equation VI.2 to calculate AUPR as shown in Algorithm VI.8. In the AUPR calculation, the denominator of each precision value is different. Thus, we need to perform division in each iteration. Since the proxy servers can determine the upper bound for numerators and denominators we can use the DIV operation to perform divisions.

VI.6 Security Analysis

We provide all semi-honest simulation-based security proofs for the MUX, MC, CMP and DIV functions defined in our framework as well as the computations of ppAURORA in the Supplement.

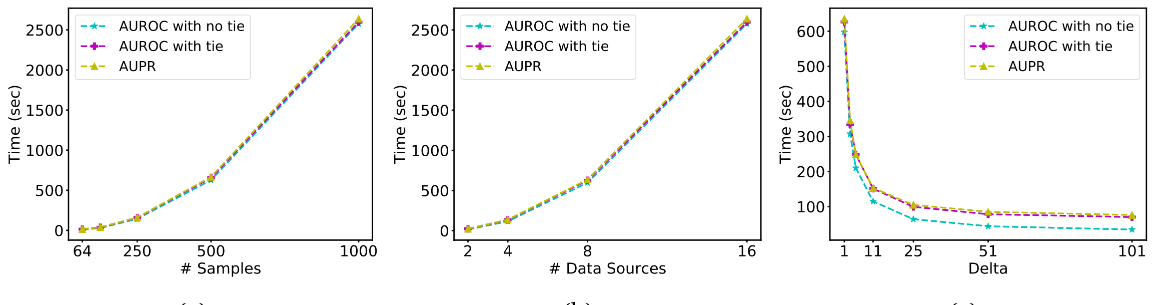


Figure VI.1: **(a)** The execution time of various settings to evaluate the scalability of ppAURORA to the number of samples for a fixed number of parties and **(b)** to the number of parties for a fixed number of samples in each party. **(c)** The effect of δ on the execution time is shown.

```
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
1  $S_0$ and $S_1$ hold a common random permutation $\pi$ for $M$ items
2  $S_0$ and $S_1$ hold a list of common random values $R$
3  $S_0$ and $S_1$ hold a list of common random permutation $\sigma$ for $\ell$ items
4  For each $i \in \{0,1\}$, $S_i$ executes Steps 5-13
5  for $j \leftarrow 1$ to $M-1$ do
6      $\langle C[j]\rangle_i \leftarrow (\langle C[j]\rangle_i - \langle C[j+1]\rangle_i)$
7      if $i = 0$ then
8          $\langle C[j]\rangle_i = L - \langle C[j]\rangle_i$
9      $\langle C[j]\rangle_i = \langle C[j]\rangle_i \oplus R[j]$
10     $\langle C[j]\rangle_i = \sigma_j(\langle C[j]\rangle_i)$
11 $\langle D\rangle_i = \pi(\langle C\rangle_i)$
12 Insert arbitrary number of dummy zero and non-zero values to randomly chosen locations in $\langle D\rangle_i$
13 $S_i$ sends $\langle D\rangle_i$ to $P_2$
14 $S_2$ reconstructs $D$ by computing $\langle D\rangle_0 \oplus \langle D\rangle_1$
15 foreach item $\langle d\rangle \in \langle D\rangle$ do
16     if $d > 0$ then
17         $d \leftarrow 1$
18 $S_2$ creates new shares of $D$, denoted by $\langle D\rangle_0$ and $\langle D\rangle_1$, and sends them to $S_0$ and $S_1$, respectively.
19 For each $i \in \{0,1\}$, $S_i$ executes Steps 18-21
20 Remove dummy zero and non-zero values from $\langle D\rangle_i$
21 $\langle C\rangle_i = \pi(\langle D\rangle_i)$
22 for $j \leftarrow 1$ to $M-1$ do
23     $\langle T[j].con\rangle_i \leftarrow \langle C[j]\rangle_i$
24 $\langle T[M].con\rangle_i \leftarrow i$
```

Algorithm VI.7: Secure detection of ties

VI.7 Dataset

To demonstrate the correctness of ppAURORA and its applicability to a real-life problem, we utilized Acute Myeloid Leukemia (AML) dataset [*] [†] from the first subchallenge of DREAM Challenge [43]. We chose the submission of the team with the lowest score in the leaderboard with accessible files, which is the team *Snail*. The training dataset has 191 samples, among which 136 patients have complete remission. We also used UCI Hearth Disease dataset [‡] for the correctness. In the test set, we have 54 samples with binary outcome.

Additionally, we also aimed to analyze the scalability of ppAURORA at different settings. For this purpose, we generated a synthetic dataset with no restriction other than having the PCVs between 0 and 1.

[*]https://www.synapse.org/#!Synapse:syn2700200

[†]https://www.synapse.org/#!Synapse:syn2501858

[‡]https://archive.ics.uci.edu/ml/datasets/heart+disease

```
input : $\langle T\rangle_i = (\{\langle con_1\rangle_i, \langle label_1\rangle_i\}, ..., \{\langle con_M\rangle_i, \langle label_M\rangle_i\})$, $\langle T\rangle_i$ is a share of the global sorted
        list of PCVs, and labels
1  $S_0$ and $S_1$ hold a common random permutation $\pi$ for $M$ items
2  For each $i \in \{0,1\}$, $S_i$ executes Steps 3-24
3  $\langle TP[0]\rangle_i \leftarrow 0$, $\langle RC[0]\rangle_i \leftarrow 0$, $\langle pPC\rangle_i \leftarrow i$, $\langle pRC\rangle_i \leftarrow 0$, $\langle N_1\rangle_i \leftarrow 0$, $\langle N_2\rangle_i \leftarrow 0$
4  for $j \leftarrow 1$ to $M$ do
5      $\langle TP[j]\rangle_i \leftarrow \langle TP[j-1]\rangle_i + \langle T[j].label\rangle_i$
6      $\langle RC[j]\rangle_i \leftarrow \langle RC[j-1]\rangle_i + i$
7  $\langle T\_TP\rangle_i = \pi(\langle TP\rangle_i)$
8  $\langle T\_RC\rangle_i = \pi(\langle RC\rangle_i)$
9  for $j \leftarrow 1$ to $M$ do
10     $\langle T\_PC[j]\rangle_i \leftarrow \mathsf{DIV}(\langle T\_TP[j]\rangle_i, \langle T\_RC[j]\rangle_i)$
11 $\langle PC\rangle_i = \pi'(\langle T\_PC\rangle_i)$
12 for $j \leftarrow 1$ to $M$ do
13     $\langle A\rangle_i \leftarrow \mathsf{MUL}(\langle pPC\rangle_i, \langle RC[j]\rangle_i - \langle pRC\rangle_i)$
14     $\langle A\rangle_i \leftarrow \mathsf{MUL}(\langle A\rangle_i, \langle T[j].con\rangle_i)$
15     $\langle N_1\rangle_i \leftarrow \langle N_1\rangle_i + \langle A\rangle_i$
16     $\langle A\rangle_i \leftarrow \mathsf{MUL}(\langle RC[j]\rangle_i - \langle pRC\rangle_i, \langle PC[j]\rangle_i - \langle pPC\rangle_i)$
17     $\langle A\rangle_i \leftarrow \mathsf{MUL}(\langle A\rangle_i, \langle T[j].con\rangle_i)$
18     $\langle N_2\rangle_i \leftarrow \langle N_2\rangle_i + \langle A\rangle_i$
19     $\langle pPC\rangle_i \leftarrow \mathsf{MUX}(\langle pPC\rangle_i, \langle PC[j]\rangle_i, \langle T[j].con\rangle_i)$
20     $\langle pRC\rangle_i \leftarrow \mathsf{MUX}(\langle pRC\rangle_i, \langle RC[j]\rangle_i, \langle T[j].con\rangle_i)$
21 $\langle N\rangle_i \leftarrow 2 \cdot \langle N_1\rangle_i + \langle N_2\rangle_i$
22 $\langle D\rangle_i \leftarrow 2 \cdot \langle TP[M]\rangle_i$
23 $\langle PRC\rangle_i \leftarrow \mathsf{DIV}(\langle N\rangle_i, \langle D\rangle_i)$
```

Algorithm VI.8: Secure AUPR computation with tie

VI.8 Results

Experimental Setup: We conducted our experiments on Amazon EC2 t2.xlarge instances. For the LAN setting, we utilized only the instances in the Frankfurt region. For the WAN setting, we additionally selected one instance from both London and Paris.

Correctness Analysis: We conducted the experiments on LAN setting. We set the precision of the division operation to cover up to the fourth decimal place. To assess the correctness of AUROC with tie, we computed the AUROC by ppAURORA and compared it to the result obtained without privacy on DREAM Challenge dataset. We obtained $AUROC = 0.693$ in both setting. To check the correctness of AUROC with no-tie of ppAURORA, we randomly picked one of the samples in DREAM Challange dataset in tie condition and generated a subset of the samples with no tie. We obtained the same AUROC with no-tie version of AUROC of ppAURORA as the non-private computation. We directly used 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. Additionally, we verified that ppAURORA computes the same AUPR as non-private computation for both DREAM Challenge and UCI dataset. AUPR scores 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. To justify the collaborative evaluation, we performed 1000 repetitions of the non-private AUROC computation with $N \in \{5, 10, 20, 40, 80, 160\}$ samples, which are randomly drawn from the whole data. The AUC starts to become more and more stable when we increase the size of the samples.

Scalability Analysis: We evaluated no-tie and with-tie versions of AUROC and AUPR of ppAURORA with $\delta = 1$ on the settings in which the number of data sources is 16 and the number of samples is $N \in \{64, 125, 250, 500, 1000\}$. The results showed that ppAURORA scales almost quadratically in terms of both communication costs among all parties and the execution time of the computation. Figure VI.1a displays the results. We also analyzed the performance of all computations of ppAURORA on a varying number of data sources. We fixed $\delta = 1$, the number of samples in each data sources to 1000 and experimented with D data sources where $D \in \{2, 4, 8, 16\}$. As Figure VI.1b summarizes, ppAURORA scales around quadratically to the number of data sources. We also analyzed the effect of $\delta \in \{1, 3, 5, 11, 25, 51, 101\}$ by fixing D to 8 and N in each data source to 1000. The execution time shown in Figure VI.1c displays 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 round trip time of WAN, which is approximately 13.2 ms. However, 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. We provided the detailed results as tables in the Supplement.

VI.9 Conclusion

In this work, we presented a novel secure 3-party computation framework and its application, ppAURORA, to compute AUC of the ROC and PR curves privately even when there exist ties in the PCVs. We proposed four novel protocols in the framework, which are MUX to select one of two secret shared values, MC to convert the ring of a secret value from $2^{\ell-1}$ to 2^{ℓ}, CMP to compare two secret shared values and DIV to divide two secret shared values. They have low round and communication complexities. The framework and its application ppAURORA are secure against passive adversaries in the honest majority setting. We implemented ppAURORA in C++ and demonstrated that ppAURORA can both compute correctly and privately, and scales quadratically to the number of parties and samples. To the best of our knowledge, ppAURORA is the first method that enables computing the exact AUC (AUROC and AUPR) privately and securely.

VI.10 Supplement

VI.10.1 Framework

In this section, we give the definitions of basic operations that we utilized in addition to the novel operations given in the main paper.

Multiplication: In two-party setting, multiplication is performed using a multiplication triple [105] which is generated via homomorphic encryption or oblivious transfer. In our 3-party setting, S_2 generates the multiplication triple and sends the shares of it to S_0 and S_1. S_0 and S_1 hold $(\langle x \rangle_0, \langle y \rangle_0)$ and $(\langle x \rangle_1, \langle y \rangle_1)$, respectively, and secure multiplication outputs fresh shares of xy (functionality $\mathcal{F}_{\mathsf{MUL}}$).

Comparison of a Secret Shared Value and a Plain Value: In this function, a value r in the ring $\mathbb{Z}_K$

```
1 Algorithm MUL()
    input  : $S_0$ and $S_1$ hold $(\langle x\rangle_0, \langle y\rangle_0)$ and $(\langle x\rangle_1, \langle y\rangle_1)$, respectively.
    output : $S_0$ and $S_1$ get $\langle z\rangle_0$ and $\langle z\rangle_1$, respectively, where $z = x \cdot y$.
2   $S_2$ picks three random numbers $a$, $b$ and $c$ where $c = a \cdot b$
3   $S_2$ generates $\langle a\rangle_i$, $\langle b\rangle_i$ and $\langle c\rangle_i$, and sends them to $S_i$ for $i \in \{0,1\}$
4   For each $i \in \{0,1\}$, $S_i$ executes Steps 5-7
5   $S_i$ computes $\langle e\rangle_i = \langle x\rangle_i - \langle a\rangle_i$ and $\langle f\rangle_i = \langle y\rangle_i - \langle b\rangle_i$
6   $S_i$ reconstructs $e$ and $f$ by exchanging the shares with $S_{1-i}$
7   $S_i$ computes $\langle z\rangle_i = i \cdot e \cdot f + f \cdot \langle x\rangle_i + e \cdot \langle y\rangle_i + \langle c\rangle_i$
```

Algorithm VI.9: Multiplication

whose bits are secret shared to S_0 and S_1 in the ring P where $P = 67$ is compared with a common value y. At the end of the secure computation, S_2 learns a bit $n' = n \oplus (r > y)$. We get the definition of this functionality $\mathscr{F}_{\mathsf{PC}}$ from [2] where it is called as Private Compare (PC). The only change that we made is S_2 sends the fresh boolean shares of n' to S_1 and S_2. PC is described in Algorithm VI.10 in the Supplement.

```
1 Algorithm PC()
    input  : $S_0, S_1$ hold $\{\langle r[j]\rangle_0^p\}_{j\in[\ell]}$ and $\{\langle r[j]\rangle_1^p\}_{j\in[\ell]}$, respectively, a common input $y$ and
             a common random bit $n$.
    output : $S_0$ and $S_1$ get $n' = n \oplus (r > y)$, $\langle n'\rangle_0^B$ and $\langle n'\rangle_1^B$, respectively.
2   $S_0, S_1$ hold $\ell$ common random values $s_j \in \mathbb{Z}_p^*$ for all $j \in [\ell]$ and a random permutation $\pi$
      for $\ell$ elements. $S_0$ and $S_1$ additionally hold $\ell$ common random values $u_j \in \mathbb{Z}_p^*$.
3   Let $t = y + 1 \bmod 2^\ell$
4   $P_i$ executes Steps $5 - 17$:
5   for $j = \ell;\ j > 0;\ j = j - 1$ do
6       if $n = 0$ then
7           $\langle w_j\rangle_i^p = \langle r[j]\rangle_i^p + iy[j] - 2y[j]\langle r[j]\rangle_i^p$
8           $\langle c_j\rangle_i^p = iy[j] - \langle r[j]\rangle_i^p + j + \sum_{k=j+1}^{\ell} \langle w_k\rangle_i^p$
9       else if $n = 1$ AND $r \neq 2^\ell - 1$ then
10          $\langle w_j\rangle_i^p = \langle r[j]\rangle_i^p + it[j] - 2t[j]\langle r[j]\rangle_i^p$
11          $\langle c_j\rangle_i^p = -it[j] + \langle r[j]\rangle_i^p + i + \sum_{k=j+1}^{\ell} \langle w_k\rangle_i^p$;
12      else
13          if $i \neq 1$ then
14              $\langle c_j\rangle_i^p = (1-i)\left(u_j + 1\right) - iu_j$
15          else
16              $\langle c_j\rangle_i^p = (-1)^j \cdot u_j$
17  Send $\{\langle d_j\rangle_i^p\}_j = \pi\left(\{s_j \langle c_j\rangle_i^p\}_j\right)$ to $P_2$
18  For all $j \in [\ell]$, $P_2$ computes $d_j = \mathsf{Reconst}(\langle d_j\rangle_0^p, \langle d_j\rangle_1^p)$ and sets $n' = 1$ iff $\exists j \in [\ell]$ such
      that $d_j = 0$.
19  $P_2$ sends $\langle n'\rangle_i^B$ to $S_i$ for $i \in \{0,1\}$
```

Algorithm VI.10: Private Compare [2]

VI.10.2 Result Tables

In this section, we provide the detailed results of the experiments with ppAURORA to compute AUROC and AUPR in Tables VI.2 and VI.3, respectively.

		Communication Costs (MB)				
M × N	δ	P_1	P_2	Helper	Total	Time (sec)
16×64	1	73.51/73.76	73.48/73.73	139.36/139.64	286.36/287.13	10.1/14.55
16×125	1	279.71/280.20	279.65/280.12	530.35/530.90	1089.71/1091.22	29.18/38.43
16×250	1	1117.32/1118.30	1117.20/1118.15	2116.65/2120.23	4351.17/4356.68	144.24/157.01
16×500	1	4466.23/4468.19	4466.00/4467.90	8469.41/8470.51	17401.65/17406.60	627.19/655.78
16×10^3	1	17858.86/17862.78	17858.4/17862.18	33863.45/33876.00	69580.71/69600.96	2578.46/2556.93
2×10^3	1	149.04/149.53	149.03/149.50	282.17/282.98	580.24/582.02	13.68/23.07
4×10^3	1	893.51/894.49	893.45/894.39	1693.99/1694.75	3480.94/3483.63	116.24/134.42
8×10^3	1	4168.03/4169.99	4167.86/4169.75	7903.60/7907.26	16239.49/16247.01	597.8/626.66
8×10^3	3	2068.78/2070.74	2068.66/2070.55	3921.74/3923.83	8059.18/8065.11	307.49/334.96
8×10^3	5	1383.59/1385.56	1383.49/1385.38	2622.98/2625.59	5390.06/5396.53	210.13/248.72
8×10^3	11	693.62/695.58	693.53/695.42	1313.98/1316.86	2701.13/2707.87	114.93/151.21
8×10^3	25	322.52/324.49	322.45/324.34	610.66/612.96	1255.63/1261.79	64.16/99.35
8×10^3	51	162.59/164.56	162.52/164.41	306.88/309.43	631.99/638.40	43.98/78.24
8×10^3	101	85.58/87.54	85.50/87.39	161.12/163.50	332.21/338.43	34.56/70.19
8×250	1	260.95/261.44	260.91/261.38	494.57/495.27	1016.43/1018.08	26.35/34.43
$8 \times UNB$	1	331.49/331.98	331.42/331.9	628.7/629.15	1291.61/1293.03	34.4/43.81

Table VI.2: The summary of the results of the experiments with ppAURORA to compute AUROC with and without tie on synthetic data. The left side of "/" represents *without-tie* results and the right side of it represents *with-tie* results. M represents the number of data sources and N represents the number of samples in one data sources. UNB represents the unbalanced sample distribution, which is $\{12, 18, 32, 58, 107, 258, 507, 1008\}$.

VI.10.3 AUC Stability Analysis

We analyzed the stability of the AUROC based on the number of test samples to justify the collaborative evaluation. We experimented with $N \in \{5, 10, 20, 40, 80, 160\}$ test samples which are randomly chosen from the DREAM Challenge Dataset. In order to have a fair evaluation, we repeated these experiments 1000 times. The experiments showed that the AUROC becomes more reliable and stable if we increase the number of test samples. In case a data source has no more additional test samples, the collaborative evaluation can be a best option. Figure VI.2 summarizes the results of the analysis.

VI.10.4 Privacy Preserving Merging

We include some sorting examples to demonstrate the process. In Figure VI.3, we show the merging of two lists of PCVs with the same size. In this example, $\delta = 1$. By this setting, we do not decrease the number of possible merging of two individually sorted lists, which can be computed as:

$$\sum_{i=0}^{|L_2|-1} \binom{|L_1|+1}{i+1} \binom{|L_2|-1}{i} \tag{VI.4}$$

M × N	δ	Communication Costs (MB)				Time (sec)
		P_1	P_2	Helper	Total	
16×64	1	73.79	73.75	139.70	287.24	15.49
16×125	1	280.25	280.17	530.90	1091.32	39.50
16×250	1	1118.39	1118.24	2120.17	4356.80	160.25
16×500	1	4468.38	4468.08	8470.66	17407.12	666.28
16×10^3	1	17863.16	17862.55	33874.20	69599.90	2639.34
2×10^3	1	149.58	149.55	283.02	582.14	23.09
4×10^3	1	894.58	894.49	1695.60	3484.67	137.44
8×10^3	1	4170.19	4169.94	7905.92	16246.05	635.72
8×10^3	3	2070.93	2070.73	3924.48	8066.14	346.11
8×10^3	5	1385.75	1385.57	2624.46	5395.77	249.01
8×10^3	11	695.77	695.61	1316.72	2708.10	152.96
8×10^3	25	324.68	324.52	613.25	1262.45	104.86
8×10^3	51	164.75	164.59	309.92	639.25	85.14
8×10^3	101	87.73	87.58	163.67	338.98	75.98
8×250	1	261.49	261.43	495.38	1018.29	35.95
$8 \times UNB$	1	332.03	331.94	629.20	1293.17	46.00

Table VI.3: The summary of the results of the experiments of AUPR computation with ppAURORA on synthetic data. M represents the number of data sources and N represents the number of samples in one data sources. UNB represents the unbalanced sample distribution, which is $\{12, 18, 32, 58, 107, 258, 507, 1008\}$.

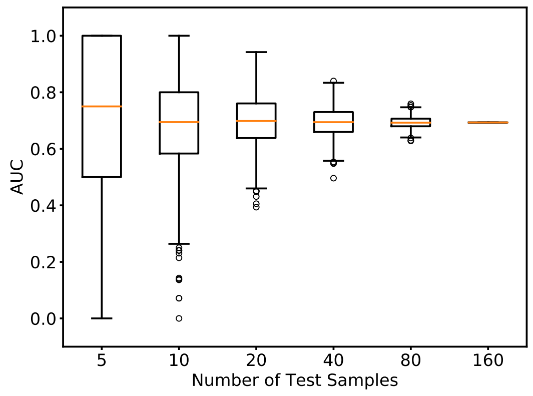


Figure VI.2: AUROC for varying number of test samples randomly chosen from the whole dataset

where L_1 and L_2 are the individually sorted lists and $|.|$ denotes the size operation.

Since the cost of having fully private merging is high and may not necessary for some applications, we can have an intermediate solution. By setting δ to a higher odd value, we can speed up the merging process. Figure VI.4 and VI.5 demonstrates an example of such a merging with $\delta = 3$. The total number of *shuffling* is 5 and 3, respectively.

In Figure VI.6, we show the sorting when the number of PCVs in the second list is 1 to justify why we set $\delta = 1$ regardless of the initial δ due to the privacy reasons. By setting δ, we secure that the

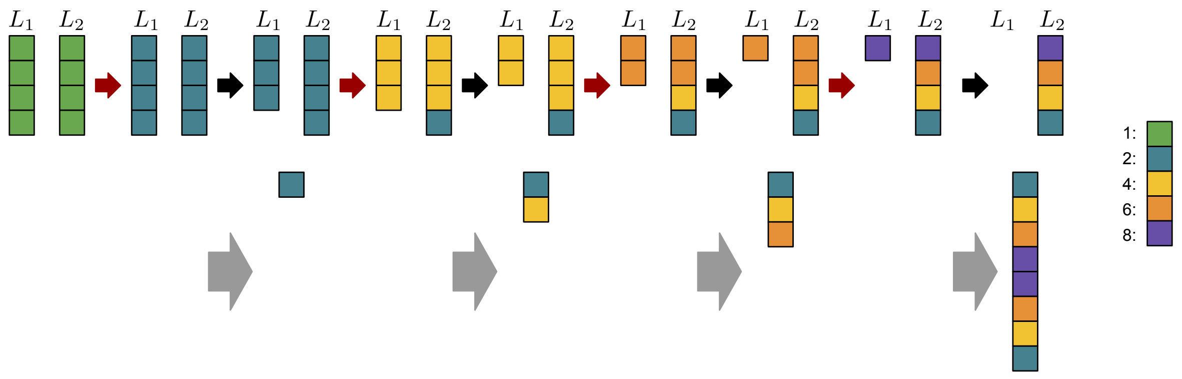


Figure VI.3: The merging of two same size lists with $\delta = 1$. The red arrows represent *shuffling*, the black ones denote *moving* the larger of the first list to the global sorted list. The progress of the global sorted list is shown by the grey arrows. Each color in the boxes represents different number of candidate PCVs. The coding of the colors are shown on the right most side of the figure. For each red arrow, i.e. for each *shuffling*, we utilize PC operation on PCVs from two lists which are on the same index to select the larger of them and employ the result of this comparison along with MUXs put the larger one into the L_1 and the other into L_2. Afterward, we *move* the top of the first list to the global sorted list. Since $\delta = 1$, we perform *shuffling* after we move the top element.

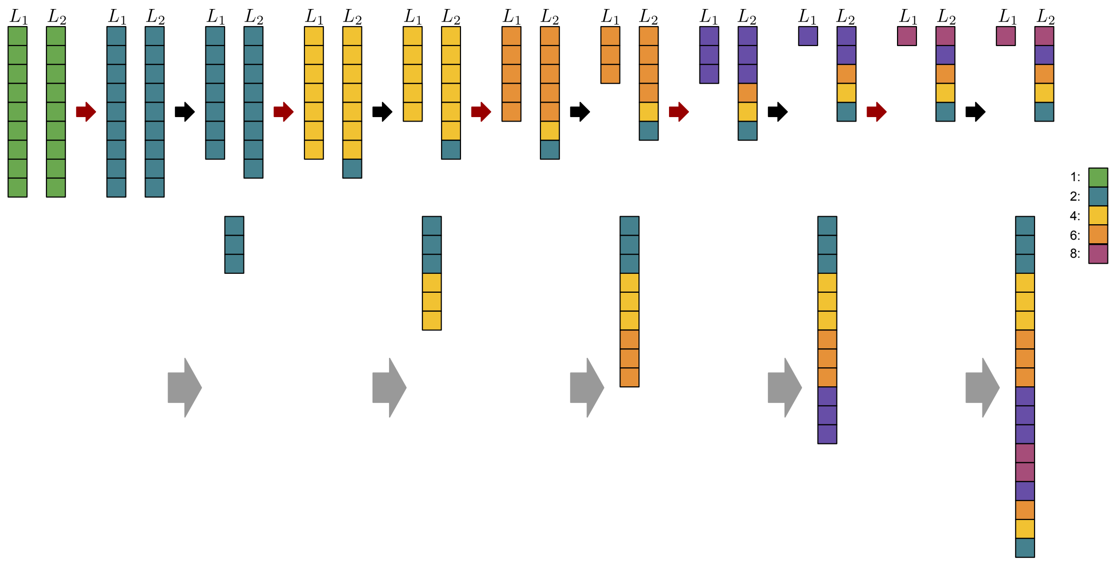


Figure VI.4: The merging of two same size lists with $\delta = 3$. We again start with *shuffling* and then move the top PCV of the first list into the global sorted list. Afterward, we compare the second element of the first list and the top element of the second list via PC. The proxies reconstruct the result of this comparison and move the share of the larger of the compared PCVs to the global sorted list. They continue until they move δ PCVs to the global sorted list. Then, they *shuffle* and repeat the same procedure until there is no PCV in the first list.

number of possible PCVs in each position of the global sorted list is the same as one can see on such a merging without privacy. The beginning and the end of the global sorted list have two possible matching, however, due to the nature of the individual sorting, the number of possible PCVs for the

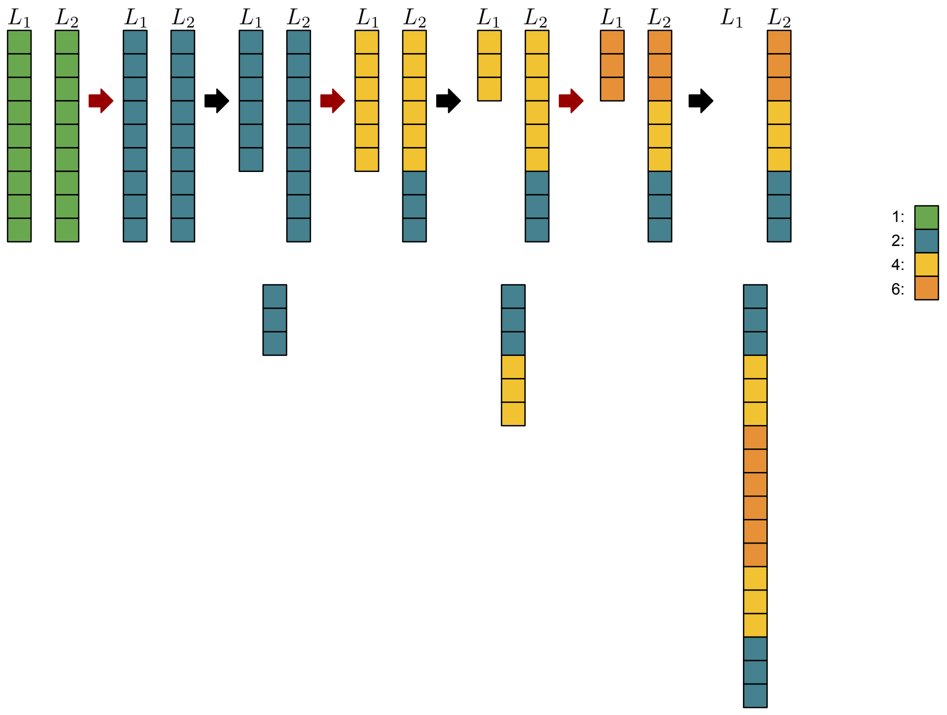


Figure VI.5: The merging of two same size lists with $\delta = 3$. This example shows how the merging happens in case all the PCVs are taken from the first list.

other positions is only 3, which is lower than the ones in Figure VI.3.

VI.10.5 Security Analysis

Here, we provide semi-honest simulation-based security proofs for the MUX, MC, CMP and DIV functions we have defined in our framework. Since the protocols we propose for AUC calculation use MUX, MC, CMP, DIV and previously defined functions, we prove the security of the main protocols in the F-hybrid model by proving the security of each function we call.

Lemma 13. The protocol MUL in Algorithm VI.9 in the Supplement securely realizes the functionality $\mathscr{F}_{\mathsf{MUL}}$.

Proof. In order to prove the correctness of our protocol we show that $\langle z\rangle_0 + \langle z\rangle_1 = xy$.

$$\begin{aligned}\langle z\rangle_0 + \langle z\rangle_1 &= f\cdot\langle x\rangle_0 + e\cdot\langle y\rangle_0 + \langle c\rangle_0 - e\cdot f\\ &\quad + f\cdot\langle x\rangle_1 + e\cdot\langle y\rangle_1 + \langle c\rangle_1\\ &= (fx + ey + c - ef)\\ &= ((y-b)x + (x-a)y + c - (x-a)(y-b))\\ &= xy - xb + xy - ya + c - xy + xb + ya - ab\\ &= xy\end{aligned} \tag{VI.5}$$

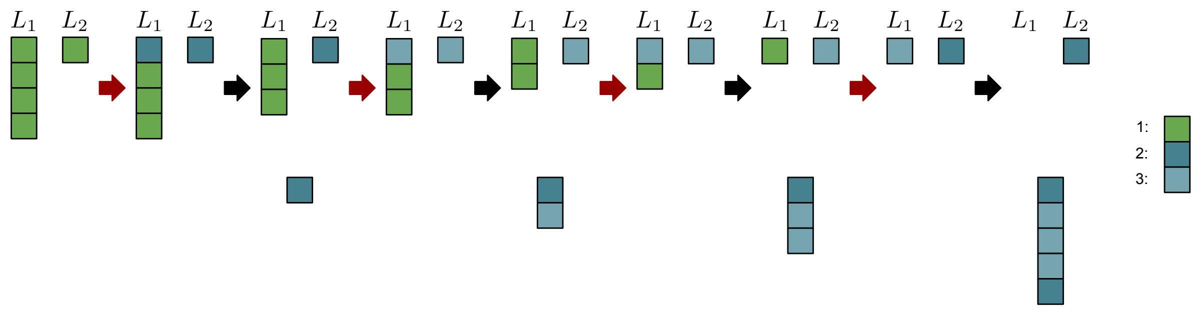


Figure VI.6: The merging of two lists with $\delta = 1$. The second list has only one PCV. The red arrows represent *shuffling*, the black ones denote *moving* the larger of the first list to the global sorted list. The progress of the global sorted list is shown by the grey arrows. Each color in the boxes represents different number of candidate PCVs. The coding of the colors are shown on the right most side of the figure.

We prove the security of our protocol. During the protocol execution, S_2 sends a multiplication triple to S_0 and S_1 and does not receive any values. Thus the view of S_2 is empty and it is very easy to prove security in case S_2 is corrupted. S_i where $i \in \{0,1\}$ sees $\langle a \rangle_i, \langle b \rangle_i, \langle c \rangle_i, \langle e \rangle_i$ and $\langle f \rangle_i$. These values are uniformly distributed random values and hence can be perfectly simulated. □

Lemma 14. The protocol MUX in Algorithm VI.1 in the main paper securely realizes the functionality $\mathscr{F}_{\mathsf{MUX}}$.

Proof. We first prove the correctness of our protocol. $\langle z \rangle_i$ is the output of S_i where $i \in \{0,1\}$. We need to prove that $\mathsf{Reconstruct}(\langle z \rangle_i) = (1-b)x + by$.

$$\begin{aligned}
\langle z \rangle_0 + \langle z \rangle_1 = {} & \langle x \rangle_0 + s - \langle b \rangle_0(\langle x \rangle_0 - \langle y \rangle_0) + r_1 \langle b \rangle_0 \\
& + r_2(\langle x \rangle_0 - \langle y \rangle_0) + r_2 r_3 + \langle x \rangle_1 + t \\
& - \langle b \rangle_1(\langle x \rangle_1 - \langle y \rangle_1) + r_0(\langle x \rangle_1 - \langle y \rangle_1) \\
& + r_0 r_1 + r_3 \langle b \rangle_1 - \langle b \rangle_0 \langle x \rangle_1 + \langle b \rangle_0 \langle y \rangle_1 \\
& - \langle b \rangle_0 r_1 - r_0 \langle x \rangle_1 + r_0 \langle y \rangle_1 - r_0 r_1 \\
& - \langle x \rangle_0 \langle b \rangle_1 - \langle x \rangle_0 r_2 + \langle y \rangle_0 \langle b \rangle_1 + \langle y \rangle_0 r_2 \\
& - r_3 \langle b \rangle_1 - r_3 r_2 - s - t \\
= {} & (1 - \langle b \rangle_0 - \langle b \rangle_1)(\langle x \rangle_0 + \langle x \rangle_1) \\
& + (\langle b \rangle_0 + \langle b \rangle_1)(\langle y \rangle_0 + \langle y \rangle_1) \\
= {} & (1-b)x + by
\end{aligned} \tag{VI.6}$$

Next we prove the security of our protocol. S_2 gets M_2, M_3, M_5 and M_6. All these values are uniformly random values because they are generated using uniformly random values r_0, r_1, r_2, r_4. S_2 computes $M_2 M_5 + M_3 M_6$. The computed value is still uniformly random because it contains uniformly random values r_0, r_1, r_2, r_4. As a result, any value learned by S_2 is perfectly simulated. For each $i \in \{0,1\}$, S_i learns a fresh share of the output. Thus S_i cannot associate the share of the output with the shares of the inputs and any value learned by S_i is perfectly simulatable. □

Lemma 15. The protocol PC in Algorithm VI.10 in the Supplement securely realizes the functionality $\mathscr{F}_{PC}$.

Proof. The proof of PC is given in [2]. □

Lemma 16. The protocol MC in Algorithm VI.2 in the main paper securely realizes the functionality $\mathscr{F}_{MC}$ in $\mathscr{F}_{\mathsf{PC}}$ hybrid model.

Proof. First, we prove the correctness of our protocol by showing $(\langle x\rangle_0^K + \langle x\rangle_1^K) \mod K = (\langle x\rangle_0 + \langle x\rangle_1) \mod L$. In the protocol, $y = (x+r) \mod K$ and $\mathsf{isWrap}(x,r,K) = r \overset{?}{>} y$, that is, $\mathsf{isWrap}(x,r,K) = 1$ if $r > y$, 0 otherwise. At the beginning, S_0, S_1 and S_2 call $\mathscr{F}_{PC}$ to compute $c = r \overset{?}{>} y$ and S_0 and S_1 obtain the boolean shares c_0 and c_1, respectively. Besides, S_2 sends also the boolean shares w_0 and w_1 of $w = \mathsf{isWrap}(\langle r\rangle_0, \langle r\rangle_1, K)$ to S_0 and S_1, respectively. If $\mathsf{isWrap}(\langle y\rangle_0, \langle y\rangle_1, K)$ is 1 then S_0 adds K to $\langle y\rangle_0$ to change the ring of y from K to L. To convert r from ring K to ring L, S_0 and S_1 add K to their shares of r based on their boolean shares w_0 and w_1, respectively. If $w_0 = 1$, then S_0 adds K to its r_1 and S_1 does the similar with its shares. Later, we need to fix is the summation of x and r, that is the value y. In case of $x + r \geq K$, we cannot fix the summation value y in ring L by simply converting it from ring K to ring L. This summation should be $x + r$ in ring L rather than $(x+r)\ mod\ K$. To handle this problem, S_0 and S_1 add K to their shares of y based on their shares c_0 and c_1. As a result, we convert the values y and r to ring L and fix the value of y if necessary. The final step to obtain x_i for party S_i is simply subtract r_i from y_i where $i \in \{0,1\}$.

Next, we prove the security of our protocol. S_2 involves this protocol in execution of $\mathscr{F}_{PC}$. We give the proof $\mathscr{F}_{PC}$ above. At the end of the execution of $\mathscr{F}_{PC}$, S_2 learns n'. However, $n' = n \oplus (x > r)$ and S_2 does not know n. Thus n' is uniformly distributed and can be perfectly simulated with randomly generated values. S_i where $i \in \{0,1\}$ sees fresh shares of $\langle r\rangle_i^K$, $\{\langle r[j]\rangle_i^p\}_{j\in[\ell]}$, w_i^B and n_i^B. These values can be perfectly simulated with randomly generated values. □

Lemma 17. The protocol CMP in Algorithm VI.3 in the main paper securely realizes the functionality $\mathscr{F}_{CMP}$ in $\mathscr{F}_{\mathsf{MC}}$ hybrid model.

Proof. First, we prove the correctness of our protocol. Assume that we have ℓ-bit number u. $v = u - (u \mod 2^{\ell-1})$ is either 0 or $2^{\ell-1}$. v is only represented with the most significant bit (MSB) of u. In our protocol, $\langle z\rangle_i$ is the output of S_i where $i \in \{0,1\}$. We need to prove that $\mathsf{Reconstruct}(\langle z\rangle_i)$ is 1 if $x < y$ and 0 otherwise. S_i where $i \in \{0,1\}$ computes $d_i^K = (x_i - y_i) \mod K$ which is a share of d over K. S_i computes d_i which is a share of d over L by invoking MC. Note that $z = x - y - \mathsf{Reconstruct}(\langle d\rangle_i)$ and all bits of z are 0 except MSB of z which is equal to MSB of $(x-y)$. Now we need to map z to 1 if it's equal to K and 0 if it's equal to 0. S_0 sends the z_0 and $z_0 + K$ in random order to S_2 and S_1 sends the z_1 to S_2. S_2 reconstructs two different values, divides these values by K, creates two additive shares of them, and sends these shares to S_0 and S_1. Since S_0 and S_1 know the order of the real MSB value, they correctly select the shares of its mapped value.

Second, we prove the security of our protocol. S_i where $i \in \{0,1\}$ sees $\langle d \rangle_i$ which is a fresh share of d and $\langle a[0] \rangle_i$ and $\langle a[1] \rangle_i$ one of them is a fresh share of the MSB of $x-y$ and the other is a fresh share of the complement of the MSB of $x-y$. Thus the view of S_i can be perfectly simulated with randomly generated values. □

Lemma 18. The protocol DIV in Algorithm VI.4 in the main paper securely realizes the functionality $\mathscr{F}_{\mathsf{DIV}}$.

Proof. We first prove the correctness of our protocol. $\langle z \rangle_i$ is the output of S_i where $i \in \{0,1\}$. We prove that $\mathsf{Reconstruct}(\langle z \rangle_i) = \frac{xF}{y}$.

$$\begin{aligned}
\langle z \rangle_0 + \langle z \rangle_1 &= \frac{(r_1\langle x \rangle_0 + r_0\langle y \rangle_0 + r_1\langle x \rangle_1 + r_0\langle y \rangle_1)F}{r_1\langle y \rangle_0 + r_1\langle y \rangle_1} - \frac{r_0F}{r_1} \\
&= \frac{(r_1x + r_0y)F}{r_1y} - \frac{r_0F}{r_1} \\
&= \frac{xF}{y} + \frac{r_0F}{r_1} - \frac{r_0F}{r_1} \\
&= \frac{xF}{y}
\end{aligned} \tag{VI.7}$$

DIV method produces correct results when S_0 and S_1 know the upper limit of x and y values. In this case, the values of r_0 and r_1 are chosen in such a way that the values $r_1\langle x \rangle_0 + r_0\langle y \rangle_0 + r_1\langle x \rangle_1 + r_0\langle y \rangle_1$ and $r_1\langle y \rangle_0 + r_1\langle y \rangle_1$ do not wrap around $\mathbb{Z}_L$. If wrapping occurs around $\mathbb{Z}_L$, DIV method produces the wrong result.

Next we prove the security of our protocol. S_2 gets a_0, a_1, b_0 and b_1. All these values are uniformly random values because they are generated using uniformly random values r_0 and r_1. As a result, any value learned by S_2 is perfectly simulated. For each $i \in \{0,1\}$, S_i learns a fresh share of the output. Thus S_i cannot associate the share of the output with the shares of the inputs and any value learned by S_i is perfectly simulatable. □

Lemma 19. The protocol in Algorithm VI.5 in the main paper securely computes AUROC in ($\mathscr{F}_{\mathsf{MUL}}$, $\mathscr{F}_{\mathsf{DIV}}$) 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. In order to compute N, we employed Equation VI.1 in the main paper. 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 Equation VI.1 in the main paper. Therefore, we can conclude that we correctly compute the AUROC.

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

Lemma 20. The protocol in Algorithm VI.7 in the main paper 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 T, $t[j].con = 0$ if $(C[j]-C[j+1]) = 0$, $t[j].con = 1$, otherwise. We first calculate the difference of successive items in C. Let assume we have two additive shares $(\langle a\rangle_0, \langle a\rangle_1)$ of a over the ring $\mathbb{Z}_L$. If $a = 0$, then $(L - \langle a\rangle_0) \oplus \langle a\rangle_1 = 0$ and if $a \neq 0$, then $(L - \langle a\rangle_0) \oplus \langle a\rangle_1 \neq 0$ where $L - \langle a\rangle_0$ is the additive modular inverse of $\langle a\rangle_0$. We use this fact in our protocol. S_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 $\langle c\rangle_0'''$. S_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 $\langle c\rangle_1'''$. S_i $i \in \{0,1\}$ 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 $\langle D\rangle_1$, S_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 S_2, S_i $i \in \{0,1\}$ removes dummy values and permutes remaining values by π'. Therefore, our protocol correctly maps items of C to 0 or 1.

We next prove the security of our protocol. S_i where $i \in \{0,1\}$ calculates the difference of successive prediction values. The view of S_2 is D which includes real and dummy zero values. S_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 differences, the index j where $D[j] = 0$, the index j where $D[j] \neq 0$ are uniformly random. The number of zero and non-zero values are not known to S_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 are hidden. Thus S_2 can not infer anything about real values in C. Furthermore, the number of repeating predictions is not known to S_2 due to a random permutation π. □

Lemma 21. The protocol in Algorithm VI.6 in the main paper securely computes AUROC in ($\mathscr{F}_{\mathsf{MUL}}$, $\mathscr{F}_{\mathsf{MUX}}$,$\mathscr{F}_{\mathsf{DIV}}$) hybrid model.

Proof. In order to compute the AUROC in case of tie, we utilize Equation VI.2 in the main paper, of which we calculate the numerator and the denominator separately. The calculation of the denominator D is the same as Lemma 19. 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 no-tie version of privacy preserving AUROC computation. This part corresponds to the rectangle areas in 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 $t.con$. $t.con = 1$ 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 into 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 $t.con$, 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 Equation VI.2 in the main paper, we multiply N_1 by 2 to make them have common denominator. Afterwards, we sum N_1 and N_2 to obtain N In order 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. S_i where $i \in \{0,1\}$ sees $\{\langle RL \rangle\}_{j\in M}$, $\{\langle A \rangle\}_{j\in M}$, $\{\langle pFP \rangle\}_{j\in M}$, $\{\langle pTP \rangle\}_{j\in M}$, $\langle D \rangle$ and $\langle ROC \rangle$ which are fresh shares of these values. Thus the view of S_i is perfectly simulatable with uniformly random values. □

Lemma 22. The protocol in Algorithm VI.8 in the main paper securely computes AUPR in ($\mathscr{F}_{\mathsf{MUL}}$, $\mathscr{F}_{\mathsf{MUX}}$,$\mathscr{F}_{\mathsf{DIV}}$) hybrid model.

Proof. In order to compute the AUPR , we utilize Equation VI.2 in the main paper of which we calculate the numerator and the denominator separately. We nearly perform the same computation with the AUROC computation in case of tie. The main difference is that we need perform 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 VI.6 in the main paper. The readers can follow the proof of Lemma 21.

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

Lemma 23. The sorting protocol in Section 5 in the main paper securely merges two sorted lists in ($\mathscr{F}_{\mathsf{CMP}}$,$\mathscr{F}_{\mathsf{MUX}}$) hybrid model.

Proof. First, we prove the correctness of our merge sort algorithm. L_1 and L_2 are two sorted lists. In the merging of L_1 and L_2, the corresponding values are first compared using the secure CMP operation. The larger values are placed in L_1 and the smaller values are placed in L_2, after the secure MUX operation is called twice. This process is called *shuffling* because it shuffles the corresponding values in the two lists. After the shuffling process, we know that the largest element of the two lists is the top element of L_1. Therefore, it is removed and added to the global sorted list L_3. On the next step, the top elements of L_1 and L_2 are compared with the CMP method. The comparison result is reconstructed by S_0 and S_1 and the top element of L_1 or L_2 is removed based on the result of CMP and added to L_3. The selection operation also gives the largest element of L_1 and L_2 because L_1 and L_2 are sorted and the selection operation selects the larger of the top elements of L_1 and L_2. 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 in ordered manner.

Next, we prove the security of our merge sort algorithm. In the shuffling operation, CMP and MUX operations 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 operations and MUX operation generates fresh shares of the corresponding values. Therefore, S_0 and S_1 cannot precisely map these values to the values in L_1 and L_2. In the selection operation, CMP is called and the selection is performed based on the reconstructed output of CMP. S_0 and S_1 are still unable to map the values added to L_3 to the values in L_1 and L_2 precisely because 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 S_0 and S_1 are perfectly simulatable with random values due to the shuffling process performed at regular intervals.

It is possible in some cases to use unshuffled values in selection operations. To prevent this, 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. $\square$

Privacy against Malicious Adversaries: Araki et al. [143] defined the notion of privacy against malicious adversaries in the client-server setting. In this setting, 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. This notion of privacy says that a malicious party cannot break the privacy of input and output of the honest parties. This setting is very similar to our setting. In our framework, two parties exchange a seed which is used to generate common randoms between them. Two parties randomize their shares using these random values which are not known 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.

Bibliography

[1] Ali Burak Ünal, Mete Akgün, and Nico Pfeifer. A framework for a fast privacy preserving calculation of non-linear kernels for machine learning applications in precision medicine. In *Proceedings of the 18th International Conference on Cryptology and Network Security CANS*, October 2019.

[2] Sameer Wagh, Divya Gupta, and Nishanth Chandran. Securenn: Efficient and private neural network training. *IACR Cryptol. ePrint Arch.*, 2018:442, 2018.

[3] How much data is created every day in 2021? [you'll be shocked!], Dec 2021. URL https://techjury.net/blog/how-much-data-is-created-every-day/.

[4] The cost of sequencing a human genome. URL https://www.genome.gov/about-genomics/fact-sheets/Sequencing-Human-Genome-cost.

[5] Muhammad Naveed, Erman Ayday, Ellen W Clayton, Jacques Fellay, Carl A Gunter, Jean-Pierre Hubaux, Bradley A Malin, and XiaoFeng Wang. Privacy in the genomic era. *ACM Computing Surveys (CSUR)*, 48(1):6, 2015.

[6] Shuang Wang, Xiaoqian Jiang, Siddharth Singh, Rebecca Marmor, Luca Bonomi, Dov Fox, Michelle Dow, and Lucila Ohno-Machado. Genome privacy: challenges, technical approaches to mitigate risk, and ethical considerations in the united states. *Annals of the New York Academy of Sciences*, 1387(1):73, 2017.

[7] Abraham P Schwab, Hung S Luu, Jason Wang, and Jason Y Park. Genomic privacy. *Clinical chemistry*, 64(12):1696–1703, 2018.

[8] Nora K Speicher and Nico Pfeifer. Integrating different data types by regularized unsupervised multiple kernel learning with application to cancer subtype discovery. *Bioinformatics*, 31(12): i268–i275, 2015.

[9] Yasin Ilkagan Tepeli, Ali Burak Ünal, Furkan Mustafa Akdemir, and Oznur Tastan. Pamogk: a pathway graph kernel-based multiomics approach for patient clustering. *Bioinformatics*, 36 (21):5237–5246, 2020.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[11] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[12] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. 2015.

[13] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[14] General Data Protection Regulation. Regulation eu 2016/679 of the european parliament and of the council of 27 april 2016. *Official Journal of the European Union*, 2016.

[15] Peter Meinicke, Maike Tech, Burkhard Morgenstern, and Rainer Merkl. Oligo kernels for datamining on biological sequences: a case study on prokaryotic translation initiation sites. *BMC bioinformatics*, 5(1):169, 2004.

[16] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

[17] Dexiong Chen, Laurent Jacob, and Julien Mairal. Recurrent Kernel Networks. *arXiv preprint arXiv:1906.03200*, 2019.

[18] Yasser EL-Manzalawy, Drena Dobbs, and Vasant Honavar. Predicting linear b-cell epitopes using string kernels. *Journal of Molecular Recognition: An Interdisciplinary Journal*, 21(4): 243–255, 2008.

[19] Saghi Nojoomi and Patrice Koehl. A weighted string kernel for protein fold recognition. *BMC bioinformatics*, 18(1):1–14, 2017.

[20] Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for svm protein classification. In *Biocomputing 2002*, pages 564–575. World Scientific, 2001.

[21] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444, 2002.

[22] Sepp Hochreiter, Martin Heusel, and Klaus Obermayer. Fast model-based protein homology detection without alignment. *Bioinformatics*, 23(14):1728–1736, 2007.

[23] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. *SIAM Journal on Computing*, 43(2):905–929, 2014.

[24] Benny Applebaum. Garbled circuits as randomized encodings of functions: a primer. In *Tutorials on the Foundations of Cryptography*, pages 1–44. Springer, 2017.

[25] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pages 162–167, Washington, DC, USA, 1986. IEEE Computer Society. ISBN 0-8186-0740-8. doi: 10.1109/SFCS.1986.25. URL https://doi.org/10.1109/SFCS.1986.25.

[26] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. ACM. ISBN 0-89791-221-7. doi: 10.1145/28395.28420. URL http://doi.acm.org/10.1145/28395.28420.

[27] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[28] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[29] Jaideep Vaidya, Hwanjo Yu, and Xiaoqian Jiang. Privacy-preserving svm classification. *Knowledge and Information Systems*, 14(2):161–178, 2008.

[30] Jaideep Vaidya and Chris Clifton. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 13(4):593–622, 2005.

[31] Jun Zhang, Xin Wang, Siu-Ming Yiu, Zoe L Jiang, and Jin Li. Secure dot product of outsourced encrypted vectors and its application to svm. In *Proceedings of the Fifth ACM International Workshop on Security in Cloud Computing*, pages 75–82. ACM, 2017.

[32] Hongchao Zhou and Gregory Wornell. Efficient homomorphic encryption on integer vectors and its applications. In *2014 Information Theory and Applications Workshop (ITA)*, pages 1–9. IEEE, 2014.

[33] Yan Liu, Pei-Yun Hsueh, Jennifer Lai, Mirweis Sangin, Marc-Antoine Nussli, and Pierre Dillenbourg. Who is the expert? analyzing gaze data to predict expertise level in collaborative applications. In *2009 IEEE International Conference on Multimedia and Expo*, pages 898–901. IEEE, June 2009. doi: 10.1109/ICME.2009.5202640.

[34] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38. IEEE, 2017.

[35] Maya Bakshi and Mark Last. Cryptornn-privacy-preserving recurrent neural networks using homomorphic encryption. In *International Symposium on Cyber Security Cryptography and Machine Learning*, pages 245–253. Springer, 2020.

[36] Thomas Lengauer, Oliver Sander, Saleta Sierra, Alexander Thielen, and Rolf Kaiser. Bioinformatics prediction of hiv coreceptor usage. *Nature biotechnology*, 25(12):1407–1410, 2007.

[37] Benedikt W Hosp, Florian Schultz, Oliver Höner, and Enkelejda Kasneci. Soccer goalkeeper expertise identification based on eye movements. *PloS one*, 16(5):e0251070, 2021.

[38] Brendan David-John, Candace Peacock, Ting Zhang, T Scott Murdison, Hrvoje Benko, and Tanya R Jonker. Towards gaze-based prediction of the intent to interact in virtual reality. In *ACM Symposium on Eye Tracking Research and Applications*, pages 1–7, 2021.

[39] Erroll Wood, Tadas Baltrušaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. Learning an appearance-based gaze estimator from one million synthesised images. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, pages 131–138, 2016.

[40] Benedict Röder, Nicolas Kersten, Marius Herr, Nora K Speicher, and Nico Pfeifer. web-rmkl: a web server for dimensionality reduction and sample clustering of multi-view data based on unsupervised multiple kernel learning. *Nucleic acids research*, 2019.

[41] Nimrod Rappoport and Ron Shamir. Multi-omic and multi-view clustering algorithms: review and cancer benchmark. *Nucleic acids research*, 46(20):10546–10562, 2018.

[42] Lifeng Zhou and Chunguang Li. Outsourcing Eigen-Decomposition and Singular Value Decomposition of Large Matrix to a Public Cloud. *IEEE Access*, 4:869–879, 2016.

[43] David P Noren, Byron L Long, Raquel Norel, Kahn Rrhissorrakrai, Kenneth Hess, Chenyue Wendy Hu, Alex J Bisberg, Andre Schultz, Erik Engquist, Li Liu, et al. A crowdsourcing approach to developing and assessing prediction algorithms for aml prognosis. *PLoS computational biology*, 12(6):e1004890, 2016.

[44] Jorge S Reis-Filho. Next-generation sequencing. *Breast Cancer Research*, 11(3):S12, 2009.

[45] Erman Ayday, Emiliano De Cristofaro, Jean-Pierre Hubaux, and Gene Tsudik. Whole genome sequencing: Revolutionary medicine or privacy nightmare? *Computer*, 48(2):58–66, 2015.

[46] Jeantine E Lunshof, Ruth Chadwick, Daniel B Vorhaus, and George M Church. From genetic privacy to open consent. *Nature Reviews Genetics*, 9(5):406, 2008.

[47] Gulce Kale, Erman Ayday, and Oznur Tastan. A utility maximizing and privacy preserving approach for protecting kinship in genomic databases. *Bioinformatics*, 34(2):181–189, 2017.

[48] Eirini Marouli, Mariaelisa Graff, Carolina Medina-Gomez, Ken Sin Lo, Andrew R Wood, Troels R Kjaer, Rebecca S Fine, Yingchang Lu, Claudia Schurmann, Heather M Highland, et al. Rare and low-frequency coding variants alter human adult height. *Nature*, 542(7640):186, 2017.

[49] Kyriaki Michailidou, Jonathan Beesley, Sara Lindstrom, Sander Canisius, Joe Dennis, Michael J Lush, Mel J Maranian, Manjeet K Bolla, Qin Wang, Mitul Shah, et al. Genome-wide association analysis of more than 120,000 individuals identifies 15 new susceptibility loci for breast cancer. *Nature genetics*, 47(4):373, 2015.

[50] Jing Ming, Eric Verner, Anand Sarwate, Ross Kelly, Cory Reed, Torran Kahleck, Rogers Silva, Sandeep Panta, Jessica Turner, Sergey Plis, et al. Coinstac: Decentralizing the future of brain imaging analysis. *F1000Research*, 6, 2017.

[51] Thomas Lengauer, Nico Pfeifer, and Rolf Kaiser. Personalized hiv therapy to control drug resistance. *Drug Discovery Today: Technologies*, 11:57–64, 2014.

[52] Fang Liu, Wee Keong Ng, and Wei Zhang. Encrypted svm for outsourced data mining. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, pages 1085–1092. IEEE, 2015.

[53] Christian Igel, Tobias Glasmachers, Britta Mersch, Nico Pfeifer, and Peter Meinicke. Gradient-based optimization of kernel-target alignment for sequence kernels applied to bacterial gene start detection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2), 2007.

[54] Britta Mersch, Alexander Gepperth, Sándor Suhai, and Agnes Hotz-Wagenblatt. Automatic detection of exonic splicing enhancers (eses) using svms. *BMC bioinformatics*, 9(1):369, 2008.

[55] Bernhard Schölkopf, Alexander J Smola, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2002.

[56] Nico Pfeifer and Oliver Kohlbacher. Multiple instance learning allows mhc class ii epitope predictions across alleles. In *International Workshop on Algorithms in Bioinformatics*, pages 210–221. Springer, 2008.

[57] Jukka-Pekka Kauppi, Melih Kandemir, Veli-Matti Saarinen, Lotta Hirvenkari, Lauri Parkkonen, Arto Klami, Riitta Hari, and Samuel Kaski. Towards brain-activity-controlled information retrieval: Decoding image relevance from meg signals. *NeuroImage*, 112:288–298, 2015.

[58] Jianguo Zhang, Kai-Kuang Ma, Meng-Hwa Er, and Vincent Chong. Tumor segmentation from magnetic resonance imaging by learning via one-class support vector machine. In *International Workshop on Advanced Image Technology (IWAIT'04)*, pages 207–211, 2004.

[59] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nc^0. *SIAM Journal on Computing*, 36(4):845–888, 2006.

[60] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *computational complexity*, 15(2):115–162, 2006.

[61] Thomas Lengauer, Oliver Sander, Saleta Sierra, Alexander Thielen, and Rolf Kaiser. Bioinformatics prediction of HIV coreceptor usage. *Nat Biotechnol*, 25(12):1407–1410, dec 2007. URL http://dx.doi.org/10.1038/nbt1371.

[62] Matthias Döring, Joachim Büch, Georg Friedrich, Alejandro Pironti, Prabhav Kalaghatgi, Elena Knops, Eva Heger, Martin Obermeier, Martin Däumer, Alexander Thielen, Rolf Kaiser, Thomas Lengauer, and Nico Pfeifer. geno2pheno[ngs-freq]: a genotypic interpretation system for identifying viral drug resistance using next-generation sequencing data. *Nucleic Acids Research*, page gky349, 2018. doi: 10.1093/nar/gky349. URL http://dx.doi.org/10.1093/nar/gky349.

[63] Angel Yu, W Lok Lai, and James Payor. Efficient integer vector homomorphic encryption, 2015.

[64] Shai Halevi and Victor Shoup. Helib-an implementation of homomorphic encryption. *Cryptology ePrint Archive, Report 2014/039*, 2014.

[65] Shai Halevi and Victor Shoup. Algorithms in helib. In *Annual Cryptology Conference*, pages 554–571. Springer, 2014.

[66] Siyuan Chen and Julien Epps. Using task-induced pupil diameter and blink rate to infer cognitive load. *Human-Computer Interaction*, 29:390–413, 2014.

[67] Tobias Appel, Christian Scharinger, Peter Gerjets, and Enkelejda Kasneci. Cross-subject workload classification using pupil-related measures. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*, ETRA '18, pages 4:1–4:8, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5706-7. doi: 10.1145/3204493.3204531.

[68] Efe Bozkir, David Geisler, and Enkelejda Kasneci. Assessment of driver attention during a safety critical situation in VR to generate VR-based training. In *ACM Symposium on Applied Perception 2019*, SAP '19, pages 23:1–23:5, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6890-2. doi: 10.1145/3343036.3343138.

[69] Thomas C. Kübler, Enkelejda Kasneci, Wolfgang Rosenstiel, Ulrich Schiefer, Katja Nagel, and Elena Papageorgiou. Stress-indicators and exploratory gaze for the analysis of hazard perception in patients with visual field loss. *Transportation Research Part F: Traffic Psychology and Behaviour*, 24:231–243, 2014.

[70] Ali Borji and Laurent Itti. Defending yarbus: Eye movements reveal observers' task. *Journal of vision*, 14, 03 2014. doi: 10.1167/14.3.29.

[71] Shahram Eivazi, Ahmad Hafez, Wolfgang Fuhl, Hoorieh Afkari, Enkelejda Kasneci, Martin Lehecka, and Roman Bednarik. Optimal eye movement strategies: a comparison of neurosurgeons gaze patterns when using a surgical microscope. *Acta neurochirurgica*, 159(6):959–966, 2017.

[72] Nora Castner, Enkelejda Kasneci, Thomas Kübler, Katharina Scheiter, Juliane Richter, Thérése Eder, Fabian Hüttig, and Constanze Keutel. Scanpath comparison in medical image reading skills of dental students: Distinguishing stages of expertise development. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*, ETRA '18, New York, NY, USA, 2018. ACM. ISBN 9781450357067. doi: 10.1145/3204493.3204550.

[73] Julian Steil and Andreas Bulling. Discovery of everyday human activities from long-term visual behaviour using topic models. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, pages 75–85, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3574-4. doi: 10.1145/2750858.2807520.

[74] Christian Braunagel, David Geisler, Wolfgang Rosenstiel, and Enkelejda Kasneci. Online recognition of driver-activity based on visual scanpath classification. *IEEE Intelligent Transportation Systems Magazine*, 9(4):23–36, 2017.

[75] Tomi Kinnunen, Filip Sedlak, and Roman Bednarik. Towards task-independent person authentication using eye movement signals. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, ETRA '10, pages 187–190, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-994-7. doi: 10.1145/1743666.1743712.

[76] Oleg V. Komogortsev, Sampath Jayarathna, Cecilia R. Aragon, and Mechehoul Mahmoud. Biometric identification via an oculomotor plant mathematical model. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, ETRA '10, pages 57–60, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-994-7. doi: 10.1145/1743666.1743679.

[77] Oleg V. Komogortsev and Corey D. Holland. Biometric authentication via complex oculomotor behavior. In *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pages 1–8. IEEE, Sep. 2013. doi: 10.1109/BTAS.2013.6712725.

[78] Yongtuo Zhang, Wen Hu, Weitao Xu, Chun Tung Chou, and Jiankun Hu. Continuous authentication using eye movement response of implicit visual stimuli. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(4):177:1–177:22, January 2018. ISSN 2474-9567. doi: 10.1145/3161410.

[79] Yasmeen Abdrabou, Mohamed Khamis, Rana Mohamed Eisa, Sherif Ismail, and Amrl Elmougy. Just gaze and wave: Exploring the use of gaze and gestures for shoulder-surfing resilient authentication. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research &*

Applications, ETRA '19, pages 29:1–29:10, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6709-7. doi: 10.1145/3314111.3319837.

[80] Shlomo Berkovsky, Ronnie Taib, Irena Koprinska, Eileen Wang, Yucheng Zeng, Jingjie Li, and Sabina Kleitman. Detecting personality traits using eye-tracking data. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 221:1–221:12, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-5970-2. doi: 10.1145/3290605.3300451.

[81] Julian Steil, Inken Hagestedt, Michael Xuelin Huang, and Andreas Bulling. Privacy-aware eye tracking using differential privacy. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*, ETRA '19, pages 27:1–27:9, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6709-7. doi: 10.1145/3314111.3319915.

[82] Daniel J. Liebling and Sören Preibusch. Privacy considerations for a pervasive eye tracking world. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, UbiComp '14 Adjunct, pages 1169–1177, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3047-3. doi: 10.1145/2638728.2641688.

[83] Julian Steil, Marion Koelle, Wilko Heuten, Susanne Boll, and Andreas Bulling. Privaceye: Privacy-preserving head-mounted eye tracking using egocentric scene image and eye movement features. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*, ETRA '19, pages 26:1–26:10, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6709-7. doi: 10.1145/3314111.3319913.

[84] Brendan John, Sanjeev Koppal, and Eakta Jain. Eyeveil: Degrading iris authentication in eye tracking headsets. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*, ETRA '19, pages 37:1–37:5, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6709-7. doi: 10.1145/3314111.3319816.

[85] Ao Liu, Lirong Xia, Andrew Duchowski, Reynold Bailey, Kenneth Holmqvist, and Eakta Jain. Differential privacy for eye-tracking data. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*, ETRA '19, pages 28:1–28:10, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6709-7. doi: 10.1145/3314111.3319823.

[86] Ali Burak Ünal, Mete Akgün, and Nico Pfeifer. A framework with randomized encoding for a fast privacy preserving calculation of non-linear kernels for machine learning applications in precision medicine. In *Cryptology and Network Security*, pages 493–511, Cham, 2019. Springer International Publishing. ISBN 978-3-030-31578-8.

[87] Erroll Wood, Tadas Baltrušaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. Learning an appearance-based gaze estimator from one million synthesised images. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, ETRA '16, page 131–138, New York, NY, USA, 2016. ACM. ISBN 9781450341257. doi: 10.1145/2857491.2857492.

[88] Seonwook Park, Xucong Zhang, Andreas Bulling, and Otmar Hilliges. Learning to find eye region landmarks for remote gaze estimation in unconstrained settings. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*, ETRA '18, pages 21:1–21:10, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5706-7. doi: 10.1145/3204493.3204545.

[89] C-A Azencott. Machine learning and genomics: precision medicine versus patient privacy. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2128):20170350, 2018.

[90] Luca Bonomi, Yingxiang Huang, and Lucila Ohno-Machado. Privacy challenges and research opportunities for genomic data sharing. *Nature Genetics*, pages 1–9, 2020.

[91] Ali Burak Ünal, Mete Akgün, and Nico Pfeifer. Escaped: Efficient secure and private dot product framework for kernel-based machine learning algorithms with applications in healthcare. *arXiv preprint arXiv:2012.02688*, 2020.

[92] Manoj M Prabhakaran and Amit Sahai. *Secure multi-party computation*, volume 10. IOS press, 2013.

[93] Yehuda Lindell. How to simulate it–a tutorial on the simulation proof technique. In *Tutorials on the Foundations of Cryptography*, pages 277–346. Springer, 2017.

[94] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[95] Cancer Genome Atlas Network et al. Comprehensive genomic characterization of head and neck squamous cell carcinomas. *Nature*, 517(7536):576–582, 2015.

[96] Antonio Aguilera and Ricardo Pérez-Aguila. General n-dimensional rotations. 2004.

[97] Tobias Sing, Valentina Svicher, Niko Beerenwinkel, Francesca Ceccherini-Silberstein, Martin Däumer, Rolf Kaiser, Hauke Walter, Klaus Korn, Daniel Hoffmann, Mark Oette, et al. Characterization of novel hiv drug resistance mutations using clustering, multidimensional scaling and svm-based feature ranking. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 285–296. Springer, 2005.

[98] Amin Allahyar and Jeroen De Ridder. FERAL: Network-based classifier with application to breast cancer outcome prediction. *Bioinformatics*, 31(12):i311–i319, 2015. ISSN 14602059. doi: 10.1093/bioinformatics/btv255.

[99] Joseph A Cruz and David S Wishart. Applications of machine learning in cancer prediction and prognosis. *Cancer informatics*, 2:117693510600200030, 2006.

[100] Payman Mohassel and Yupeng Zhang. SecureML: A System for Scalable Privacy-Preserving Machine Learning. *Proceedings - IEEE Symposium on Security and Privacy*, pages 19–38, 2017. ISSN 10816011. doi: 10.1109/SP.2017.12.

[101] Payman Mohassel and Peter Rindal. Aby^3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 35–52, 2018. doi: 10.1145/3243734.3243760. URL https://doi.org/10.1145/3243734.3243760.

[102] Thore Graepel, Kristin E. Lauter, and Michael Naehrig. ML confidential: Machine learning on encrypted data. In *Information Security and Cryptology - ICISC 2012 - 15th International Conference, Seoul, Korea, November 28-30, 2012, Revised Selected Papers*, pages 1–21, 2012. doi: 10.1007/978-3-642-37682-5_1. URL https://doi.org/10.1007/978-3-642-37682-5_1.

[103] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 201–210, 2016. URL http://proceedings.mlr.press/v48/gilad-bachrach16.html.

[104] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Cryptodl: Deep neural networks over encrypted data. *CoRR*, abs/1711.05189, 2017. URL http://arxiv.org/abs/1711.05189.

[105] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 420–432, 1991. doi: 10.1007/3-540-46766-1_34. URL https://doi.org/10.1007/3-540-46766-1_34.

[106] Hwanjo Yu, Jaideep Vaidya, and Xiaoqian Jiang. Privacy-preserving SVM classification on vertically partitioned data. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3918 LNAI:647–656, 2006. ISSN 03029743. doi: 10.1007/11731139_74.

[107] Jaideep Vaidya, Hwanjo Yu, and Xiaoqian Jiang. Privacy-preserving SVM classification. *Knowledge and Information Systems*, 14(2):161–178, 2008. ISSN 02191377. doi: 10.1007/s10115-007-0073-7.

[108] Jun Zhang, Xin Wang, Siu-Ming Yiu, Zoe L. Jiang, and Jin Li. Secure Dot Product of Outsourced Encrypted Vectors and its Application to SVM. pages 75–82, 2017. doi: 10.1145/3055259.3055270.

[109] Sameer Wagh, Divya Gupta, and Nishanth Chandran. SecureNN: 3-Party Secure Computation for Neural Network Training. *Proceedings on Privacy Enhancing Technologies*, 2019(3):26–49, 2019. doi: 10.2478/popets-2019-0035.

[110] Andrew C Yao. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*, pages 160–164. IEEE, 1982.

[111] Ali Burak Ünal, Mete Akgün, and Nico Pfeifer. A framework for a fast privacy preserving calculation of non-linear kernels for machine learning applications in precision medicine. In *Cryptology and Network Security - 17th International Conference, CANS 2019, Fuzhou, China, October 25 - 27, 2019, Proceedings*, 2019.

[112] Harry Chandra Tanuwidjaja, Rakyong Choi, and Kwangjo Kim. A survey on deep learning techniques for privacy-preserving. In *Machine Learning for Cyber Security - Second International Conference, ML4CS 2019, Xi'an, China, September 19-21, 2019, Proceedings*, pages 29–46, 2019. doi: 10.1007/978-3-030-30619-9_4. URL https://doi.org/10.1007/978-3-030-30619-9_4.

[113] Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. (February):8–11, 2015. doi: 10.14722/ndss.2015.23113.

[114] Thomas Schneider and Oleksandr Tkachenko. Episode&#58; efficient privacy-preserving similar sequence queries on outsourced genomic databases. In *Proceedings of the 2019 ACM*

Asia Conference on Computer and Communications Security, Asia CCS '19, pages 315–327, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6752-3. doi: 10.1145/3321705.3329800. URL http://doi.acm.org/10.1145/3321705.3329800.

[115] Erman Pattuk, Murat Kantarcioglu, Huseyin Ulusoy, and Bradley Malin. Cheapsmc: A framework to minimize secure multiparty computation cost in the cloud. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 285–294. Springer, 2016.

[116] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. A systematic approach to practically efficient general two-party secure function evaluation protocols and their modular design. *Journal of Computer Security*, 21:283–315, 2013.

[117] Michael O. Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005:187, 1981.

[118] Shunjie Han, Cao Qubo, and Han Meng. Parameter selection in svm with rbf kernel function. In *World Automation Congress 2012*, pages 1–4. IEEE, 2012.

[119] Thomas Schneider and Oleksandr Tkachenko. Episode: Efficient privacy-preserving similar sequence queries on outsourced genomic databases. ASIACCS, 2019.

[120] Seny Kamara and Mariana Raykova. Secure outsourced computation in a multi-tenant cloud. In *IBM Workshop on Cryptography and Security in Clouds*, pages 15–16, 2011.

[121] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. URL http://www.ietf.org/rfc/rfc5246.txt. Updated by RFCs 5746, 5878, 6176.

[122] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[123] Si Chen, Anmin Fu, Jian Shen, Shui Yu, Huaqun Wang, and Huaijiang Sun. Rnn-dp: A new differential privacy scheme base on recurrent neural network for dynamic trajectory privacy protection. *Journal of Network and Computer Applications*, 168:102736, 2020.

[124] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Cryptodl: Deep neural networks over encrypted data. *arXiv preprint arXiv:1711.05189*, 2017.

[125] Ran Gilad-Bachrach, 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*, pages 201–210. PMLR, 2016.

[126] Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. Crypten: Secure multi-party computation meets machine learning. *arXiv preprint arXiv:2109.00984*, 2021.

[127] Sameer Wagh, Divya Gupta, and Nishanth Chandran. SecureNN: 3-Party Secure Computation for Neural Network Training. *Proc. Priv. Enhancing Technol.*, 2019(3):26–49, 2019.

[128] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*, pages 643–662. Springer, 2012.

[129] Ali Burak Ünal, Nico Pfeifer, and Mete Akgün. ppAUC: Privacy Preserving Area Under the Curve with Secure 3-Party Computation. *CoRR*, abs/2102.08788, 2021. URL https://arxiv.org/abs/2102.08788.

[130] Alexey G Murzin, Steven E Brenner, Tim Hubbard, and Cyrus Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of molecular biology*, 247(4):536–540, 1995.

[131] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. {GAZELLE}: A low latency framework for secure neural network inference. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1651–1669, 2018.

[132] Payman Mohassel and Peter Rindal. Aby3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 35–52, 2018.

[133] Ivan Damgård, Daniel Escudero, Tore Frederiksen, Marcel Keller, Peter Scholl, and Nikolaj Volgushev. New primitives for actively-secure mpc over rings with applications to private machine learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1102–1120. IEEE, 2019.

[134] Megha Byali, Harsh Chaudhari, Arpita Patra, and Ajith Suresh. Flash: fast and robust framework for privacy-preserving machine learning. *Proceedings on Privacy Enhancing Technologies*, 2020 (2):459–480, 2020.

[135] Arpita Patra and Ajith Suresh. BLAZE: blazing fast privacy-preserving machine learning. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.

[136] Kamalika Chaudhuri and Staal A Vinterbo. A stability-based validation procedure for differentially private machine learning. In *Advances in Neural Information Processing Systems*, pages 2652–2660, 2013.

[137] Kendrick Boyd, Eric Lantz, and David Page. Differential privacy for classifier evaluation. In *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, pages 15–23, 2015.

[138] Yan Chen, Ashwin Machanavajjhala, Jerome P Reiter, and Andrés F Barrientos. Differentially private regression diagnostics. In *ICDM*, pages 81–90, 2016.

[139] Jacob Whitehill. How does knowledge of the auc constrain the set of possible ground-truth labelings? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5425–5432, 2019.

[140] Gregory J Matthews and Ofer Harel. An examination of data confidentiality and disclosure issues related to publication of empirical roc curves. *Academic radiology*, 20(7):889–896, 2013.

[141] Yehuda Lindell. How to simulate it - A tutorial on the simulation proof technique. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 277–346. Springer International Publishing, 2017. doi: 10.1007/978-3-319-57048-8_6. URL https://doi.org/10.1007/978-3-319-57048-8_6.

[142] Ran Canetti. 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*, pages 136–145. IEEE Computer Society, 2001. doi: 10.1109/SFCS.2001.959888. URL https://doi.org/10.1109/SFCS.2001.959888.

[143] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. 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*, pages 805–817, 2016.

[144] Benny Applebaum. Garbled circuits as randomized encodings of functions: a primer. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 1–44. Springer International Publishing, 2017. doi: 10.1007/978-3-319-57048-8_1. URL https://doi.org/10.1007/978-3-319-57048-8_1.