

RESEARCH ARTICLE

A Reliable Application of MPC for Securing the Tri-Training Algorithm

HENDRA KURNIAWAN¹, (Member, IEEE), AND MASAHIRO MAMBO², (Member, IEEE)

¹Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa 920-1192, Japan

²Institute of Science and Engineering, Kanazawa University, Kanazawa 920-1192, Japan

Corresponding author: Hendra Kurniawan (hendra@umrah.ac.id)

The work of Hendra Kurniawan was supported in part by the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan; and in part by the JICA Innovative Asia Program through Kanazawa University, Japan.

ABSTRACT Due to the widespread use of distributed data mining techniques in a variety of areas, the issue of protecting the privacy of sensitive data has received increasing attention in recent years. Privacy-preserving distributed data mining (PPDDM) focuses on decentralized data analysis without the disclosure of sensitive information from data owner. However, the previous PPDDM mostly works on a limited amount of labeled data. In contrast to the real world, unlabeled data is abundance and labeled data is scarce. The objectives of this paper are to study and to analyze privacy-preserving properties of semi-supervised learning (SSL) algorithm with the combination of labeled and unlabeled data, where data is distributed among multiple data owners. In this paper we propose a Privacy-preserving Distributed Data Mining (PPDDM) method by designing a reliable application of secure MPC to semi-supervised tri-training algorithms. We simulate the original tri-training algorithm and tri-training algorithm with secure MPC using a different types of classifiers and datasets. The simulation results show that tri-training in secure MPC has almost same accuracy compared to original tri-training algorithm. We also compare execution time in addition to performance evaluation of tri-training in secure and the original tri-training algorithms.

INDEX TERMS Distributed data mining, multi-party computation, privacy-preserving, semi-supervised learning, tri-training.


I. INTRODUCTION

Nowadays, we have known an unprecedented surge in data, especially for personal data in the business and health care environment. Most companies and organizations have private big data that are involved in medical services, social media applications, e-commerce, online banking, and interdisciplinary research data. The entire data infrastructure is built on a distributed system nationally or globally. Distributed data mining offers intriguing possibilities for fresh insights and a wide range of applications but is often fraught with concerns regarding model accuracy and data privacy. Consider the case of analyzing patient data in health care. Hospitals use patient data to improve diagnosis accuracy and efficiency [1].

However, learning patient data from a single hospital limits the performance of the model and can lead to incomplete

knowledge discovery. Individual behavior and social conditions also have a role to affect patient health [2]. Combining diverse patient data from multiple hospital sources provides a path for obtaining more accurate and reliable health outcome analysis models [3], [4], [5]. Due to the widespread use of distributed data mining techniques in a variety of areas, the issue of protecting the privacy of sensitive data has received increasing attention in recent years [6], [7].

Privacy-Preserving Distributed Data Mining (PPDDM) focuses on distributed data analysis without leaking sensitive information from one party to another. PPDDM technology aims to make it technically or mathematically impossible to derive the original data from communication messages and even from the final analysis results [7], [8]. There are three issues in PPDDM for real-world applications. That is, how to deal with some data issues (classification, regression, etc.), how to point out opposition concerns related to the data party (malicious, honest, etc.), and a balance between privacy

The associate editor coordinating the review of this manuscript and approving it for publication was Long Wang .

and model performance. Currently, several privacy protection techniques are available for distributed data mining. These include data perturbation, local learning and global integration, and secure multi-party computation (MPC) [9].

Data perturbation protects privacy by adding noise to individual datasets while retaining essential summary information [10], [11]. One approach to data perturbation is to use statistical techniques to replace original data with synthetic values that have similar statistics. Synthetic data can be generated using statistical models trained on the original data. Another approach is to distort the data using additive noise, multiplying noise, or other randomization methods [12]. Among PPDDM techniques, data perturbation is relatively simple [11]. However, the quality and accuracy of the training model can be affected. Another PPDDM approach is local learning and global integration. This approach integrates local models into a global model using an ensemble learning technique to improve performance [9]. Each party can train their own local data, which is then integrated to create centralized or global data mining to produce the final result. The original data of each party is not shared. The most well-known and widely used PPDDM is secure MPC, an encryption-based method that allows multiple parties to collectively compute functions on their own data without sharing it [6], [7]. Secure MPC also protects participants from external attacks and from each other. The result is accurate, and each party only sees the data obtained from the published result. In real-world data, secure MPC supports fixed-point and floating-point operations with controlled linear complexity [13], [14].

However, the previous PPDDM mostly works on the limited amount of labeled data [4], [12], [14], [15], [16]. In the real-world, unlabeled data is abundant, and labeled data is scarce. Semi-supervised learning (SSL), which leverages both labeled and unlabeled data, is a type of machine learning method between supervised and unsupervised learning that takes full advantage of large unlabeled samples and labeled samples. For example, for classification issues, additional data points with unknown labels can be used to support the classification process [17]. SSL training assumes that the labeled training data is supplemented by additional unlabeled data that needs to be used properly to improve the accuracy of the model [18], [19].

The disagreement-based semi-supervised classification realizes the utilization of unlabeled data by using multiple classifiers. In the learning process, unlabeled data is used as a platform for information interaction between several classifiers. The original disagreement-based algorithm was developed in [20] which is called co-training which requires two views of sufficient and redundant. Later, Goldman and Zhou proposed a new co-training method called statistical co-training [21], which uses two different learning algorithms based on a single view. In [22], Zhou and Li developed a tri-training that does not require sufficient and excessive views. Tri-training also does not require the use of a different

TABLE 1. Different types of PPDDM.

	Supervised PPDDM	Semi-supervised PPDDM
MPC-based	Liu et al. [6] Tran et al. [12] Sotthiwat et al. [16]	Proposed MPC-based Tri-training Algorithm
Non-MPC-based	Zhao et al. [23] Han et al. [24]	Fierimonte et al. [18] Li and Li [25]

supervised learning algorithm that hypothetically divides the instant space into a set of equivalent classes.

The existing work on [6], [12], and [16] propose MPC-based on supervised PPDDM to enable multiple parties to jointly train a model on a labeled dataset without sharing their private data. The focus is on protecting the privacy of the data owners while still allowing for collaborative model training. In semi-supervised PPDDM, the goal of MPC is to train a model on a dataset that contains both labeled and unlabeled data. The challenge is to make use of the unlabeled data to improve model performance without compromising the privacy of the data owners. Secure MPC can be used in semi-supervised PPDDM to allow parties to jointly train a model on the labeled data while keeping the unlabeled data private.

Combining tri-training algorithms with existing secure MPC protocols is a potential solution to achieving privacy preservation in tri-training algorithms. However, it is important to note that the presence of unlabeled data in tri-training introduces additional privacy concerns that need to be addressed. These concerns include the potential for information leakage through the use of semi-supervised learning and the need to ensure that the privacy guarantees provided by secure MPC are maintained even when dealing with unlabeled data.

Additionally, the scalability of the approach is also a challenge, especially when dealing with large datasets and complex models. Therefore, careful consideration of the specific requirements and constraints of the problem at hand is necessary to determine the most appropriate solution.

In this paper we propose a Privacy-preserving Distributed Data Mining (PPDDM) method by designing a reliable application of secure MPC to semi-supervised tri-training algorithms. In summary, the main contributions of this paper are as follows:

- 1) We introduce privacy-preserving distributed data mining implementation with semi-supervised learning algorithm. Secure multi-party computation is used to preserve data input among multiple parties and tri-training algorithm as the base of semi-supervised learning. As far as we know, this is the first study to implement and analyze the accuracy of semi-supervised tri-training algorithm using secure MPC.
- 2) We carry out a number of experiments from small to a large real-world and artificial datasets to demonstrate the reliability of the proposed design.

- 3) We record key performance metrics, which are classification accuracy and execution time to verify the effectiveness and efficiency of our design.

The rest of this paper is structured as follows. Section I gives the outline of the related work. Section III describes the preliminary of secure multi-party computation and semi-supervised learning. Section IV explains the methodology of the proposed solution for privacy-preserving semi-supervised tri-training algorithm. Furthermore, the experiment results and discussion are described in Section V. Finally, Section VI gives conclusion and provides some future work.

II. RELATED WORK

Table 1 presents different types of supervised and semi-supervised PPDDM related to MPC-based and non-MPC-based privacy preservation approach. Relatively few investigations have been conducted on semi-supervised PPDDM, and none of them utilize secure-MPC as a privacy preservation method.

Non-MPC-based supervised PPDDM was proposed by [23] and [24]. Zhao et al., [23] presented a non-MPC-based supervised PPDDM. They provided a framework to protect private training data against leakage from gradient sharing. This framework was designed by adopting a new knowledge transfer technique, known as private aggregation of teacher ensembles (PATE) in an efficient and novel manner. The framework allows effective transfer of relational knowledge from sensitive data to public data in a privacy-preserving way and enables participants to jointly learn local models based on the public data with noise-preserving labels. Han et al., [24] proposed a verifiable federated learning scheme that supports privacy protection over deep neural networks. To ensure the confidentiality of the user's local gradients, a double-masking protocol was used in the scheme. The authors used aggregation technology of data tags to ensure that the information returned by cloud server can be verified.

A non-MPC-based fully decentralized semi-supervised learning privacy-preservation was proposed by Fierimonte et al. [18]. The main component of the proposed algorithm consists of a fully distributed computation of the adjacency matrix. They extend a Laplacian regularized least square algorithm, a member of manifold regularization family. The distributed semi-supervised algorithm is efficient and scalable for low-rank distributed matrix completion, based on the framework of diffusion adaptation. The algorithm can preserve privacy by the inclusion of flexible privacy-preserving mechanisms for similarity computation. Nevertheless, in this algorithm, the whole dataset is loaded once at a time and processed in each iteration, which causing higher workload. Besides, this algorithm calculates the Euclidian Distance Matrix (EDM) with respect to total samples. Obviously, this process will consume much time and waste computational resources, especially when the scale of training data is large.

Another non-MPC-based implementation on graph-based semi-supervised PPDDM was introduced by Li and Li [25].

They proposed inductive semi-supervised learning with harmonic anchor mixture in PPDDM. The idea lies in combining mixture models and graph-based methods to construct an anchor mixture with the ability of label prediction. The authors also proposed an optimization process, which is accurately calculated through secure protocols, to achieve effectiveness. The possibility of privacy leaks from each party's input data is one of the limitations of this method, where each side shares its data in its unprocessed condition.

A PPDDM based on secure-MPC was proposed by Liu et al. [6]. They designed a high performance algorithm based on optimized matrix computation with one-hot encoding and LU decomposition for regression problem (e.g. least squares method). The authors introduced two real situations to preserve privacy.

Another an MPC-based approach in PPDDM was proposed by Tran et al. [12]. They develop a new efficient framework for privacy-preserving deep learning models. The framework is able to operate in a decentralized network environment that does not require a trusted third-party server, while maintaining the privacy of local data at a low cost of transmission bandwidth. The framework ensures not only the confidentiality of user data, but also the applicability of the paradigm in a decentralized network with no third-party server. The authors employed a model sharing strategy to prevent the direct leakage of local data and devised a secure model sharing protocol to provide secure sharing and aggregation for collaborative model construction. Furthermore, the secure model sharing protocol combines randomization techniques with the secure sum protocol. Without trusting each other or a third-party server, parties utilizing the honest-but-curious approach are protected from both inside and outside threats.

Sothiwat et al. [16] offered an alternative application of MPC to provide privacy protection for distributed supervised learning. Instead of applying heavy MPC over the entire local models for secure model aggregation, they proposed to encrypt a critical portion of model parameters (gradients) to reduce communication cost, while maintaining MPC's privacy-preserving benefits without compromising the accuracy of the jointly learned model. In particular, only the first layer of local models is encrypted with MPC technique, while the remainder is transmitted directly to the centralized node. Such a technique effectively minimizes the additional computing and communication cost caused by MPC, while maintaining MPC's advantages in terms of privacy protection and model accuracy. Even though these studies employ a dependable secure-MPC privacy protection, they have limitations in overcoming real-world challenges in data mining, particularly those involving the usage of labeled and unlabeled data. To overcome the limitations of previous work, we provide a privacy-preserving for tri-training algorithm using secure-MPC. Our work exploits the real-world problem of distributed data mining using labeled and unlabeled data, and encrypts the data to enhance privacy-preservation.

III. PRELIMINARIES

A. SECURE MULTI-PARTY COMPUTATION

Secure multi-party computation is an area of cryptography which deals with two or more parties $P_i (i = 1, 2, \dots, n)$ compute a function $f(x_1, x_2, \dots, x_n) = (y_1, y_2, \dots, y_n)$ on their private input x_i in a distributed computing environment. After the computation is complete, each side of P_i must obtain its own corresponding output y_i without obtaining any other information. The previous technique for evaluating secure functionality was for all parties to communicate their input to a trusted third party. This trustworthy party then calculates the function and returns the result to the parties. However, we wish to eliminate this trustworthy third party entirely. Intuitively, a multiparty computation is secure if the revealed information is identical to what it would have been if the computation had been conducted by encrypting communications to a trusted third party.

Basic protocols for secure MPC often fall into one of two different groups [26]. The first is based on Yao's [27] concept of a garbled circuit or Yao circuit: in this scenario, one party gives the function to be calculated as a binary circuit, while another party encrypts the gates of this circuit to generate the garbled circuit. This technique is plainly predicated on a computational assumption, namely that the encryption system is secure. The second technique is based on secret sharing schemes: suppose a party expresses the function to be calculated as an arithmetic circuit. To achieve absolute security, this second technique employs a perfectly secure secret sharing system. It turns out that the first technique appears to be more suited to the situation of two parties, whilst the second approach appears to be better suited to the case of three or more parties. In 1982, Yao [27] explicitly presented and solved the secure two-party computation issue. Goldreich et al. [28] expanded and extended the two-party computation situation to the MPC problem. The study in [13] has created and implemented a computation framework and protocol called SPDZ based on the notion and theory of multi-party computation and secret sharing. Through secret sharing and homomorphic encryption, SPDZ may perform any combination of addition and multiplication operations.

The SPDZ protocol [29] divides a MPC work into two parts. The first step is an offline preprocessing phase in which numerical triples are generated and shared among parties in advance. By using relatively homomorphic encryption, it is possible to improve speed while also guarding against a dishonest majority. The second step is an online phase in which real computation are performed. The computational cost during the actual computation of the online phase is minimal when an offline pretreatment step is used to offload work. This capability of secure multi-party computation has the potential to enable a variety of machine learning applications that are currently infeasible because of data privacy concerns. A flexible software framework that aims to make modern secure MPC techniques accessible to machine learning has been developed [30] named CRYPTEN. This framework provides a comprehensive tensor-computation

library in which all computations are performed via secure MPC. The framework design adopted the two main principles: machine-learning first API design, and Eager execution of an imperative programming model.

B. SEMI-SUPERVISED LEARNING

The main principle of semi-supervised learning is to construct learner to mark unlabeled samples using the model hypothesis in the distribution of data. To compensate for the absence of labeled data, the current semi-supervised classification uses large amounts of unlabeled data to enlarge the training set of the classification algorithm. Examples of semi-supervised classification are generative methods, semi-supervised support vector machine methods, graph-based methods and disagreement-based methods.

Generative based semi-supervised learning methods generate labeled and unlabeled examples by the same parametric model. The model parameters connect unlabeled samples and the learning target in a direct manner. Algorithm in this methods often treat the labels of unlabeled data as missing model parameter values and employ the EM (expectation-maximization) technique for maximum likelihood estimation of the model parameters. The algorithms are distinguished by the generative models used to fit the data, such as mixture of Gaussian [31], mixture of experts [32], and Naive Bayes [33]. This method is simple and straightforward to construct. Generative approaches may outperform discriminative models, e.g semi-supervised support vector machine (S3VM), when learning from a relatively small number of labeled examples. Nonetheless, approaches in this domain are severely lacking. In other words, when the model assumption is erroneous, training the model with a large number of unlabeled data would degrade performance and need more computational time.

Semi-supervised support vector machine (S3VM) methods attempt to use unlabeled data to change the decision boundary learned from a small number of labeled examples so that it explores the less dense region while maintaining the accurate classification of labeled data. This methods are based on the separation of low density. Transductive Support Vector Machine (TSVM) is the most well-known S3VM [34]. This algorithm builds an SVM using labeled samples before assigning potential labels to unlabeled data. Then, it maximizes the margin across both labeled and unlabeled data with their prospective labels by reversing the labels of the unlabeled samples on opposite sides of the decision boundary. The best solution is reached when the decision boundary not only classifies the labeled data as precisely as possible, but also avoids passing through the high density region. In the S3VM, an SVM must be trained by solving a quadratic programming problem during every learning process. As a result, its time complexity is substantial.

Graph-based semi-supervised learning methods essentially attempt to create a graph with all the data [35]. They utilize nonnegative weights on the edge of any two samples to characterize their similarity, and propagate labels from labeled

samples to unlabeled ones based on the pairwise similarity. Zhou et al. [36] suggested a regularization paradigm for directed graphs that takes the directionality and global relationship into account. To generate a graph with all training samples requires a significant amount of time.

Disagreement-based semi-supervised learning methods are based on discrepancies of multiple classifiers to classify unlabeled data during the learning process. Among the numerous semi-supervised learning methods, the disagreement-based semi-supervised learning methods are simple and effective. As stated in [37], they may perform well in many routine situations. Meanwhile, it is not always the cases in other semi-supervised learning methods. For example the three methods (generative methods, semi-supervised support vector machine methods, graph-based methods) may degrade performance or spend more computational time, and they are not suitable for secure-MPC scheme. We will focus on disagreement-based semi-supervised learning in our implementation. Well-known examples of disagreement-based semi-supervised learning are co-training [20], tri-training [22], and tri-training with disagreement [38].

Blum and Mitchell [20] proposed co-training in 1998, which is the original disagreement-based algorithm, also known as the standard simultaneous training algorithm. They assume that the dataset have two views with sufficient redundancy to meet the following criteria: First, each view is sufficient for training a strong learner. Second, the views are conditionally independent of one another.

In most real-world applications, a data set has only one attribute set, not two. As a result, typical joint training is ineffective and inefficient. Methods that do not rely on the availability of two perspectives have been developed to take advantage of learner interaction when using unlabeled data. Contrary to the findings of previous studies [20], [21], [33], Zhou and Li [22] introduced tri-training, which does not need the availability of two perspective learning algorithms. Tri-training can be used for a broader range of real-world issues. Using three classifiers, tri-training performs classification by utilizing unlabeled data. This kind of configuration addresses the difficulty of knowing how to effectively choose the most strongly anticipated unlabeled cases to label and generates a final hypothesis.

Sogaard [38] proposed tri-training with disagreement as an enhancement to the tri-training algorithm 1. The enhancement concentrated on strengthening tri-training algorithm by changing lines 16-17 with equation (1). The condition to update value of third classifier c_3 is satisfied when two classifiers c_1 and c_2 agree and the third classifier c_3 is disagrees. Tri-training with disagreement is substantially more efficient than tri-training since it imports less unlabeled data. Due to the ease of implementation in real-world problems by using only one learning algorithm to generate the three classifiers, we will adopt tri-training and tri-training with disagreement for applying MPC to disagreement-based semi-supervised

learning.

$$\text{if } c_m(x) = c_s(x) \neq c_i(x) (m, s \neq i) \text{ then} \\ \times L_i \leftarrow L_i \cup (x, c_m(x)) \quad (1)$$

In general, tri-training algorithm works as follows.

- 1) The three classifiers were first trained from the labeled data. To diversify the three classifiers, tri-training bootstraps the label samples set to generate three unique classifiers. Throughout the tri-training procedure, the three classifiers are upgraded.
- 2) In every learning round, if two classifiers agree on how to classify an unlabeled instance but a third disagrees, the third classifier will be updated using values of the first and second classifiers.
- 3) Finally, the combination of three classifiers is determined by majority vote. When the classifier no longer changes, the process stops.

Algorithm 1 describes the pseudocode of the tri-training algorithm, where L is original Labeled example set, U is Unlabeled example set, and $Learn$ is the learning algorithm (e.g. Decision Tree, Multi-layer Perceptron, Naïve Bayes, and Support Vector Machine). The output of the algorithm is the classifier c with majority voting.

IV. METHODOLOGY

A. PROBLEM DEFINITION

In a distributed system, data is distributed across multiple sites. The goal of a PPDDM approach is to compute a model utilizing all the data without any site disclosing sensitive information. In this section, we firstly describe the problem under consideration as describe in Table 2. Original dataset D is the real dataset used during simulation, then D is splitted into two parts D_{tr} and D_{tx} for training and testing process respectively. The distributed system is consisted of n Data Owner: $\{DO_1, DO_2, \dots, DO_n\}$ and m Computing Parties: $\{CP_1, \dots, CP_m\}$. Training dataset D_{tr} are horizontally distributed among the DO . Different DO may have several instances with the same set of attributes. Throughout the training, no information about each specific instance is exposed to anyone other than its owner. We assume a semi-honest adversary design, which means that the data owner and computing parties must strictly follow the PPDDM protocol specification.

B. SECURE MPC FOR TRI-TRAINING ALGORITHM

Secure MPC enables participants to compute data while keeping it secret. This secure MPC capability has the potential to enable a wide range of machine learning applications that are now infeasible due to data privacy issues. This section describes how tri-training algorithm as a part of semi-supervised classification works on a secure MPC system.

Secure MPC can be used in a variety of ways. Essentially secure MPC is used in a distributed computing environment. Figure 1 shows the architecture of secure MPC in a distributed system with n Data Owner DO and m Computing Parties

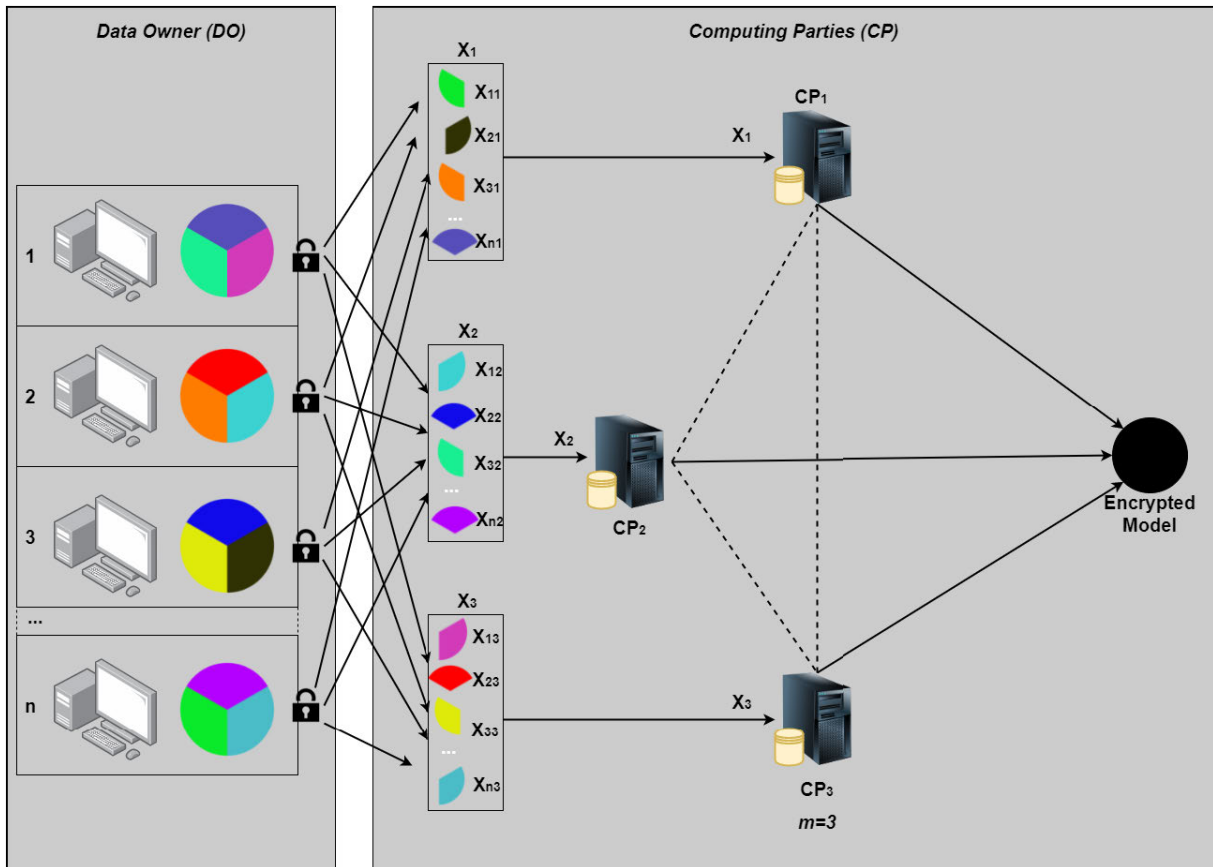


FIGURE 1. Architecture of secure MPC with n data owner (DO) and m computing parties (CP) with encrypted dataset.

TABLE 2. Notation used in the system.

Notation	Description
D	Original dataset $D = D_{tr} \cup D_{tx}$
D_{tr}	Training dataset
D_{tx}	Testing dataset
X	Encrypted training dataset $X = \bar{L} \cup \bar{U}$
L	Original labeled dataset
\bar{L}	Encrypted labeled dataset
U	Original unlabeled dataset
\bar{U}	Encrypted unlabeled dataset
DO	Data Owner $\{DO_1, \dots, DO_n\}$
CP	Computing Parties $\{CP_1, \dots, CP_m\}$

CP using encrypted dataset. Each Data Owner DO holds D_{tr} and shares the same key k , then encrypts $X = Enc_k(D_{tr})$ to produce encrypted data X from D_{tr} , where $Enc_k(\cdot)$ is an additive homomorphic encryption [39] with key k . The X from each DO is sent to CP , then the CP will provide secure computation of tri-training algorithm from X 's and produce the encrypted model $\bar{\epsilon}$.

The following assumptions are stated in this application: privacy-preservation mode uses the “semi-honest” adversary [40] (also referred to as the “honest-but-curious” threat model). Every party follows the protocol exactly: it executes

all of the calculations stated in the program and delivers the proper results to the parties mentioned in the program. The communication channel is secure: no party has access to data that has not been directly disclosed to it. Every side has a private source of randomness. Parties may utilize any previously viewed data and undertake arbitrary processing to infer information.

Two semi-supervised learning algorithm Tri-training [22] and Tri-training with disagreement [38] is used in secure MPC process based on Algorithm 1. The basic idea of this implementation is to compute both the tri-training algorithm and tri-training with disagreement algorithm using secure MPC for each Computing Party CP with encrypted dataset input from multiple Data Owner DO .

Suppose we have n data owner $\{DO_1, \dots, DO_n\}$ that ready to contribute their data and m computing parties $\{CP_1, \dots, CP_m\}$ to perform secure MPC in this application. We provide original dataset D during experiment, then split D into two part D_{tr} and D_{tx} based on *split ratio* for training and testing purpose. In our implementation we use 0.8 and 0.2 ratio for D_{tr} and D_{tx} respectively. Training dataset D_{tr} is splitted into n part $\{D_{tr-1}, \dots, D_{tr-n}\}$. Then $\{D_{tr-1}, \dots, D_{tr-n}\}$ is assigned to each $\{DO_1, \dots, DO_n\}$ respectively.

Algorithm 1 Tri-Training Algorithm

```

1: Input:  $L$ : Original labeled example set
2:  $U$ : Unlabeled example set
3:  $Learn$ : Learning algorithm
4: Output: Classifier  $c$ 
5: for  $i \in \{1, 2, 3\}$  do
6:    $S_i \leftarrow BootstrapSample(L)$ 
7:    $c_i \leftarrow Learn(S_i)$ 
8:    $e'_i \leftarrow 0.5; l'_i \leftarrow 0$ 
9: end for
10: repeat
11:   for  $i \in \{1, 2, 3\}$  do
12:      $L_i \leftarrow 0; update_i \leftarrow FALSE$ 
13:      $e_i \leftarrow MeasureError(c_j \& c_k)(j, k \neq i)$ 
14:     if ( $e_i < e'_i$ ) then
15:       for each  $x \in U$  do
16:         if  $c_m(x) = c_s(x)(m, s \neq i)$  then
17:            $L_i \leftarrow L_i \cup (x, c_m(x))$ 
18:         end if
19:       end for
20:       if  $l'_i = 0$  then
21:          $l'_i \leftarrow \frac{e_i}{e'_i - e_i} + 1$ 
22:       end if
23:       if ( $l'_i < |L_i|$ ) then
24:         if ( $e_i |L_i| < e'_i l'_i$ ) then
25:            $update \leftarrow TRUE$ 
26:         else if  $l'_i > \frac{e_i}{e'_i - e_i}$  then
27:            $L_i \leftarrow Subsample(L_i, \frac{e'_i l'_i}{e_i})$ 
28:            $update \leftarrow TRUE$ 
29:         end if
30:       end if
31:     end if
32:   end for
33:   for  $i \in \{1, 2, 3\}$  do
34:     if ( $e_i |L_i| < e'_i l'_i$ ) then
35:        $c_i \leftarrow Learn(L \cup L_i); e'_i \leftarrow e_i; l'_i \leftarrow |L_i|$ 
36:     end if
37:   end for
38: until none of  $c_i (i \in \{1, 2, 3\})$  changes
39:  $c(x) \leftarrow \arg \max_{y \in label} \sum_{i: c_i(x)=y} 1$ 

```

Firstly, each DO encrypts its D_{tr} training dataset $X = Enc_k(D_{tr})$, where X consisted of $\{\bar{L} \cup \bar{U}\}$ and then split X into m shares $\{X_1, \dots, X_m\}$ using random masks, 3 random shares in this example. These random shares can be combined to reconstruct the original X dataset. These shares $\{X_1, X_2, X_3\}$ are then distributed between 3 CP $\{CP_1, CP_2, CP_3\}$ using secret sharing. In this process, encrypted shares are distributed in random order. As a result, none of the CP know which shares they hold. The same computing procedure is executed for tri-training and tri-training with disagreement algorithm as described in algorithm 2:

Algorithm 2 Secure MPC for Tri-Training Algorithm With m Computing Parties $\{CP_1, \dots, CP_m\}$

```

1: Input: Encrypted labeled dataset  $\{\bar{L}_1, \dots, \bar{L}_m\}$ ,
2: Encrypted unlabeled dataset  $\{\bar{U}_1, \dots, \bar{U}_m\}$ ,
3:  $TrainClassifier$ : Train  $\bar{L}$  using learning algorithm.
4: Output: Encrypted model  $\bar{c}$ 
5:  $CP_1 \leftarrow (\bar{L}_1 \cup \bar{U}_1), \dots, CP_m \leftarrow (\bar{L}_m \cup \bar{U}_m)$ 
6: for  $i \in \{1, 2, 3\}$  do
7:   for each  $CP$  do
8:      $S_i \leftarrow BootstrapSample(\bar{L})$ 
9:      $\bar{c}_i \leftarrow TrainClassifier(S_i)$ 
10:   end for
11: end for
12: repeat
13:   for  $i \in \{1, 2, 3\}$  do
14:     for  $x \in U$  do
15:       if  $\bar{c}_m(x) = \bar{c}_s(x)(m, s \neq i)$  then
16:          $\bar{L}_i \leftarrow \bar{L}_i \cup (x, \bar{c}_m(x))$ 
17:       end if
18:     end for
19:      $\bar{c}_i \leftarrow TrainClassifier(\bar{L} \cup \bar{L}_i)$ 
20:   end for
21: until none of  $\bar{c}_i (i \in \{1, 2, 3\})$  changes
22: apply majority vote over  $\bar{c}_i$ 

```

- 1) CP_1, CP_2, CP_3 hold secret shares X_1, X_2, X_3 respectively, where $X_1 = \bar{L}_1 \cup \bar{U}_1, X_2 = \bar{L}_2 \cup \bar{U}_2$, and $X_3 = \bar{L}_3 \cup \bar{U}_3$.
- 2) Encrypted labeled dataset \bar{L} for each CP can be defined as: $\bar{L}_1 = (x_{11}, y_{11}), \dots, (x_{1l}, y_{1l}), \bar{L}_2 = (x_{21}, y_{21}), \dots, (x_{2j}, y_{2j})$, and $\bar{L}_3 = (x_{31}, y_{31}), \dots, (x_{3\kappa}, y_{3\kappa})$.
- 3) For each CP , setup $S_i \leftarrow BootstrapSample(\bar{L})$ for $i \in \{1, 2, 3\}$.
- 4) For each $CP, \bar{c}_i \leftarrow TrainClassifier(S_i)$: for $i \in \{1, 2, 3\}$ using learning algorithm.
- 5) Training process stops when none of $\bar{c}_i, i \in \{1, 2, 3\}$ change.
- 6) Final result is an *EncryptedModel* based on the majority vote over \bar{c}_i .

Each CP jointly computes all functions according to the secure MPC protocol in CRYPTEN framework [30] during simulation. The result is jointly computed and then can be distributed (encrypted) to the user as a service. Finally, we calculate the classification accuracy (equation 2) of *Encrypted model* \bar{c} with an input testing dataset D_{tx} .

V. RESULT AND DISCUSSION

In this section, we discuss the experimental environment and the results. We conducted experiments using six datasets with four learning algorithms as classifiers for tri-training algorithm. Using original training input data D_{tr} and encrypted private training input data X , we compare the performance of secure MPC for tri-training and tri-training with

TABLE 3. Public dataset used in privacy-preserving semi-supervised learning.

No	Dataset	Samples	Features	Classes
1	AUSTRALIAN	690	14	2
2	WDBC	569	30	2
3	MNIST	60,000	784	10
4	UNSW-NB15	2,540,044	49	9
5	HIGGS	11,000,000	28	2
6	CIC-IDS	15,450,706	78	7

disagreement algorithm. The simulation for original input data is performed without privacy-preservation whereas the simulation process for private input data is executed under the privacy-preservation mode using secure MPC. When comparing cryptographic techniques, performance factors such as classification accuracy and execution time should be considered [11].

A. EXPERIMENTAL SETUP

1) SIMULATION ENVIRONMENT

We provide the following computing environment for the experiments. A server computer has a CPU configuration of Intel Xeon Bronze 3206R 1.90GHz, 32.0 GB RAM, storage 2 TB and run under Ubuntu-20.04 operating system. The software specification to implement the simulation for original tri-training and tri-training with disagreement are conda version 4.10.3, python 3.9.7, torch 1.10.11, torchvision 0.11.2, omegaconf 2.1.1, onnx 1.10.2, pandas 1.3.5, and tensorboard 2.8.0. The experiments on data classification for tri-training and tri-training with disagreement using secure multi-party computation are carried out under the CRYPTEN framework [30].

2) DATASETS USED IN THE EXPERIMENTS

The datasets for the experiment are public classification datasets that have been widely used in the information security and machine learning research communities. The dimensions of the datasets used for the performance testing vary from small to extremely large, to check the performance of the proposed application in different circumstances. Table 3 describes the number of samples and descriptions of each dataset. The first one is AUSTRALIAN credit card application dataset, which consists of 690 samples and 14 features with 2 classes. This small dataset is intriguing due to its mixed characteristics, nominal with a limited number of values and nominal with a higher number of values. The second one is Breast Cancer Wisconsin Diagnostic dataset (WDBC) contains 569 samples, 30 features, and 2 classes to conclude whether they are benign or malignant cancer diagnostic [41]. The next is MNIST dataset of handwritten digits formatted as 28×28 images, with 784 features created [42]. This dataset is composed of 60,000 examples with 10 label decisions to recognize a one digit number between 0-9. The UNSW-NB15 dataset [43] is used in this experiment, which has 2,540,044 rows of data with 49 attributes and 9 classifications. The

Australian Centre for Cyber Security's (ACCS) Cyber Range Lab created the UNSW-NB15 dataset as a mixture of genuine modern regular activities and synthetic contemporary assault behaviors. One of the huge datasets used in this experiment is HIGGS dataset [44], which has 11,000,000 records. The HIGGS dataset has 28 attributes and 2 labels for a classification problem produced using Monte Carlo simulations. Finally, CIC-IDS [45] is the largest dataset used in this experiment, providing 15,450,706 samples and 78 features. The CIC-IDS dataset includes 7 distinct attack scenarios, such as brute-force, heartbleed, botnet, DoS, DDoS, online assaults, and network penetration.

We also provide artificial dataset for simulation with different dimensionality, noise intensity, extreme outliers, and non-uniform distribution aspect. Table 5 shows artificial datasets with noise aspect [46]. We use MNIST handwritten dataset generated using Gaussian noise. We vary the variance of Gaussian noise in the MNIST dataset to evaluate the classification accuracy of the original tri-training algorithm and tri-training with disagreement algorithm. In figure 2 we show the Gaussian noise in MNIST handwritten dataset. Figure 2a represents the original MNIST handwritten dataset, figure 2b presents the Gaussian noise in MNIST handwritten dataset with mean 0.0 and variance 0.05, figure 2c shows the MNIST handwritten dataset with Gaussian noise with mean 0.0 and variance 0.1, figure 2d displays the MNIST handwritten dataset with Gaussian noise with mean 0.0 and variance 0.5, and finally, figure 2e displays the MNIST handwritten dataset with Gaussian noise with mean 0.0 and variance 0.7.

Table 6 displays non-uniform distribution aspect for artificial dataset [47], [48]. The Forest cover type dataset is a non-uniform distribution dataset with 581,012 records, 55 features, and 7 classes. The Epileptic Sizable Recognition dataset is also a non-uniform dataset with 11,500 records, 178 features, and 5 classes.

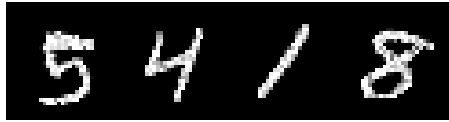
In table 7, we provide artificial dataset with different dimensionality. We generate four different artificial datasets with $10,000 \times 50$ dimensions, $10,000 \times 100$ dimensions, $10,000 \times 150$ dimensions, and $10,000 \times 200$ dimensions. Finally, in table 8 we generate two extreme outliers with $10,000 \times 5$ dimensions and $10,000 \times 10$ dimensions.

The Inter quartile range *IQR* technique is utilized to identify the extreme outliers of a given dataset. This method divides the dataset into quartiles, with the first quartile Q_1 representing the value below which 25% of the data points, the second quartile Q_2 being the median point of the dataset, and the third quartile Q_3 indicating the value below which 75% of the data points. The *IQR* value can be obtained by calculating $IQR = Q_3 - Q_1$.

In Table 8, we provide artificial datasets containing extreme outlier values. These datasets differ in the total number of extreme outlier values, with the first dataset having 2, the second dataset having 11, and the third dataset having 24 such values. Extreme outliers are those data points that fall outside the range of $Q_1 - 3 * IQR$ or $Q_3 + 3 * IQR$. Figure 3 demonstrates an example of visualizing the



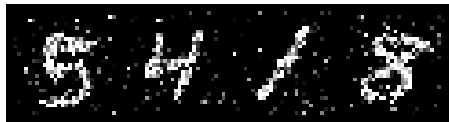
(a) Example of original MNIST handwritten digit images.



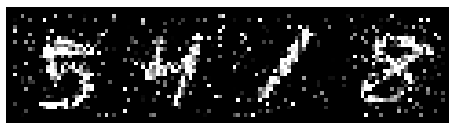
(b) Gaussian noise in MNIST handwritten digit images with mean 0.0 and variance 0.05.



(c) Gaussian noise in MNIST handwritten digit images with mean 0.0 and variance 0.1.



(d) Gaussian noise in MNIST handwritten digit images with mean 0.0 and variance 0.5.



(e) Gaussian noise in MNIST handwritten digit images with mean 0.0 and variance 0.7.

FIGURE 2. MNIST handwritten digit image with additional gaussian noise.

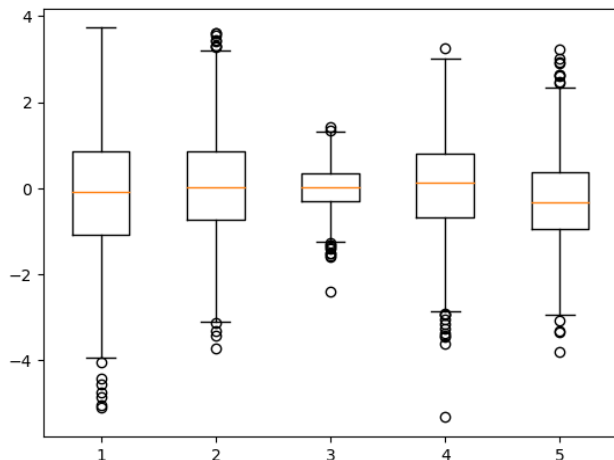


FIGURE 3. Visual representation displaying boxplot chart of extreme outlier data point from artificial dataset.

extreme outlier dataset using a boxplot chart that includes 5 features.

3) CLASSIFIER USED IN THE EXPERIMENTS

Typically, a classification model is used to predict the class label for an unlabeled data object. In this experiment,

we employ various classification techniques, including the decision tree, naive bayes, multi-layer perceptron (MLP), and support vector machines (SVM) classifiers, to support tri-training and tri-training with disagreement algorithm. Each technique employs distinct learning algorithms to construct models with excellent generalizability. In general, the classification model is constructed by dividing the available dataset into a training set, which is used during the classifier’s construction phase, and a test set for validation.

- Decision tree classifiers provide a readable classification model that is potentially accurate in many different application contexts. The decision tree classifier [49] creates the classification model by building a decision tree. Each node in the tree specifies a test for an attribute, and each branch descending from that node corresponds to one of the attribute’s possible values. Each leaf represents the associated class labels for the instance. Instances in the training set are classified by navigating them from the tree’s root to a leaf, based on the results of tests along the path. Each node, beginning with the root node of the tree, divides the instance space into two or more sub-spaces based on an attribute test condition. Then, a new node is created by moving down the tree branch corresponding to the attribute’s value. This procedure is then repeated for the subtree rooted at the new node until all training set records have been categorized.
- The Naive Bayes classifiers [50] are a family of Bayesian classifiers. The number of parameters required by Naive Bayes classifiers is proportional to the number of variables (features/predictors) in a learning problem. This experiment utilizes Gaussian Naive Bayes for numerical/continuous characteristics. Assume that the distribution of continuous values is Gaussian. Consequently, likelihood probabilities are calculated using the Gaussian distribution.
- MLP [51] is a learning algorithm that trains on a dataset to learn a function. MLP is capable of learning a non-linear function approximator for classification or regression. At least three layers of nodes comprise an MLP: an input layer, a hidden layer, and an output layer. Each node, excluding input nodes, is a neuron with a nonlinear activation function. MLP classifier employs backpropagation, a supervised learning technique, for training.
- SVMs were first proposed in statistical learning theory [49]. SVM can deal with high-dimensional data and generates a very comprehensive (geometric) model. An SVM predictor is based on the kernel function K , which defines a specific type of similarity measure between data objects. Kernel functions include linear, RBF (radial basis function), polynomial, and sigmoid kernels, among others. The SVM learning problem can be formulated as a convex optimization problem, in which a variety of algorithms can be used to locate the global minimum of the objective function.

B. PERFORMANCE EVALUATION

We evaluated the performance of secure MPC for tri-training and tri-training with disagreement regard to classification accuracy and execution time. Classification accuracy is calculated based on the fraction of correctly classified samples in equation (2) from original D_{tx} data testing to encrypted model from proposed model, the best performance is 1. While execution time is calculated for the whole training process, execution time is measured in second. All measurement is calculated in average value based on 10-fold cross validation [52]. The *label ratio* is a ratio of $\frac{L}{D_{tr}}$ for original tri-training and tri-training with disagreement algorithm, and ratio of $\frac{L}{X}$ for tri-training in secure MPC and tri-training disagreement in secure MPC.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (2)$$

$$CompAccTriTrain = \frac{AccTriTrain}{AccTriTrainMPC} \quad (3)$$

$$CompAccTriDisagr = \frac{AccTriDisagr}{AccTriDisagrMPC} \quad (4)$$

where FP , TN , TP and FN definitions are the false positive, true negative, true positive, and false negative, respectively. We define $CompAccTriTrain$ in equation (3) as the comparison of classification accuracy between tri-training in secure MPC and original tri-training algorithm, $CompAccTriDisagr$ in equation (4) as the comparison of classification accuracy between tri-training with disagreement in secure MPC and original tri-training with disagreement algorithm.

1) CLASSIFICATION ACCURACY FOR WIDELY USED DATASETS

Figure 4 shows classification accuracy for the AUSTRALIAN dataset with different classifiers and the effect of increasing *label ratio*. Figure 4(a) describes classification accuracy for decision tree classifier. Both tri-training and tri-training in secure MPC have almost equal accuracy with different values of *label ratio*. The same indication show for original tri-training with disagreement and tri-training with disagreement in secure MPC. Figure 4(b) shows comparison of classification accuracy using Naive Bayes classifier. The accuracy value of tri-training to tri-training in secure MPC is almost the same for different value of *label ratio*, the same evidence show for accuracy comparison of tri-training with disagreement to tri-training with disagreement using secure MPC. Figure 4(c) describes classification accuracy for AUSTRALIAN dataset using MLP classifier. For all *label ratio*, we get the same accuracy for tri-training and tri-training using MPC, and almost the same accuracy for tri-training with disagreement and tri-training with disagreement using MPC. Figure 4(d) shows the accuracy for SVM classifier. For all classifiers we can see the accuracy is increasing while *label ratio* is increasing, this indicates the large portion of L labeled data, we can achieve better accuracy. We calculate the average value of $CompAccTriTrain$ and $CompAccTriDisagr$ for each classifier and *label ratio*, both $CompAccTriTrain$

and $CompAccTriDisagr$ have the same value 0.99. We can conclude that tri-training in secure MPC and tri-training with disagreement in secure MPC achieve the same performance accuracy as the original tri-training and tri-training with disagreement algorithm.

Figure 5 illustrates the classification accuracy for the WDBC dataset using several classifiers and the effect of changing the *label ratio*. Figure 5(a) depicts the classification precision of the decision tree classifier. With varying *label ratio* values, tri-training and tri-training in secure MPC achieve nearly identical levels of accuracy. Both original tri-training with disagreement and tri-training with disagreement in secure MPC display the same indication. Figure 5(b) illustrates a comparison in classification precision using the Naive Bayes classifier. The accuracy value of tri-training to tri-training in secure MPC is nearly same for different *label ratio* values; the same is true for tri-training with disagreement to tri-training with disagreement in secure MPC. Figure 5(c) demonstrates the classification accuracy for the WDBC dataset using the MLP classifier. For each label ratio, tri-training and tri-training with MPC yield identical accuracy, while tri-training with disagreement and tri-training with disagreement with MPC yield almost identical accuracy. Figure 5(d) illustrates the precision of the SVM classifier. We can see that the accuracy of each classifier improves as the proportion of L labeled data increases. Calculating the average value of $CompAccTriTrain$ and $CompAccTriDisagr$ for each classifier and *label ratio* yields the same value for both variables: 1.00. Tri-training in secure MPC and tri-training with disagreement in secure MPC attain the same accuracy in performance as the original tri-training and tri-training with disagreement algorithm.

Figure 6 depicts the classification accuracy of the MNIST dataset using several classifiers, as well as the effect of varying the *label ratio*. Only Naive Bayes classifier have the accuracy under 0.90 in figure 6(b). Decision tree classifier has the best accuracy 1.00 for all *label ratio* in figure 6(a). We compute the average variable value of $CompAccTriTrain$ and $CompAccTriDisagr$ for each classifier and *label ratio*, the same outcome for both variables is 1.00. Tri-training in secure MPC and tri-training with disagreement in secure MPC achieve the same performance precision as the original tri-training and tri-training with disagreement algorithm.

The next experiment is executed on large dataset UNSW-NB15, HIGGS and CIC-IDS. Tabel 4 display the classification accuracy of original tri-training, tri-training with disagreement, tri-training in secure MPC, tri-training with disagreement in secure MPC. The classification accuracy reaches maximum value in each dataset with different classifiers and label ratios. This indicate that, both tri-training in secure MPC and tri-training with disagreement in secure MPC achieve best performance for large dataset.

2) CLASSIFICATION ACCURACY FOR ARTIFICIAL DATASETS

We evaluate the original tri-training and tri-training in secure computation, tri-training with disagreement, and tri-training

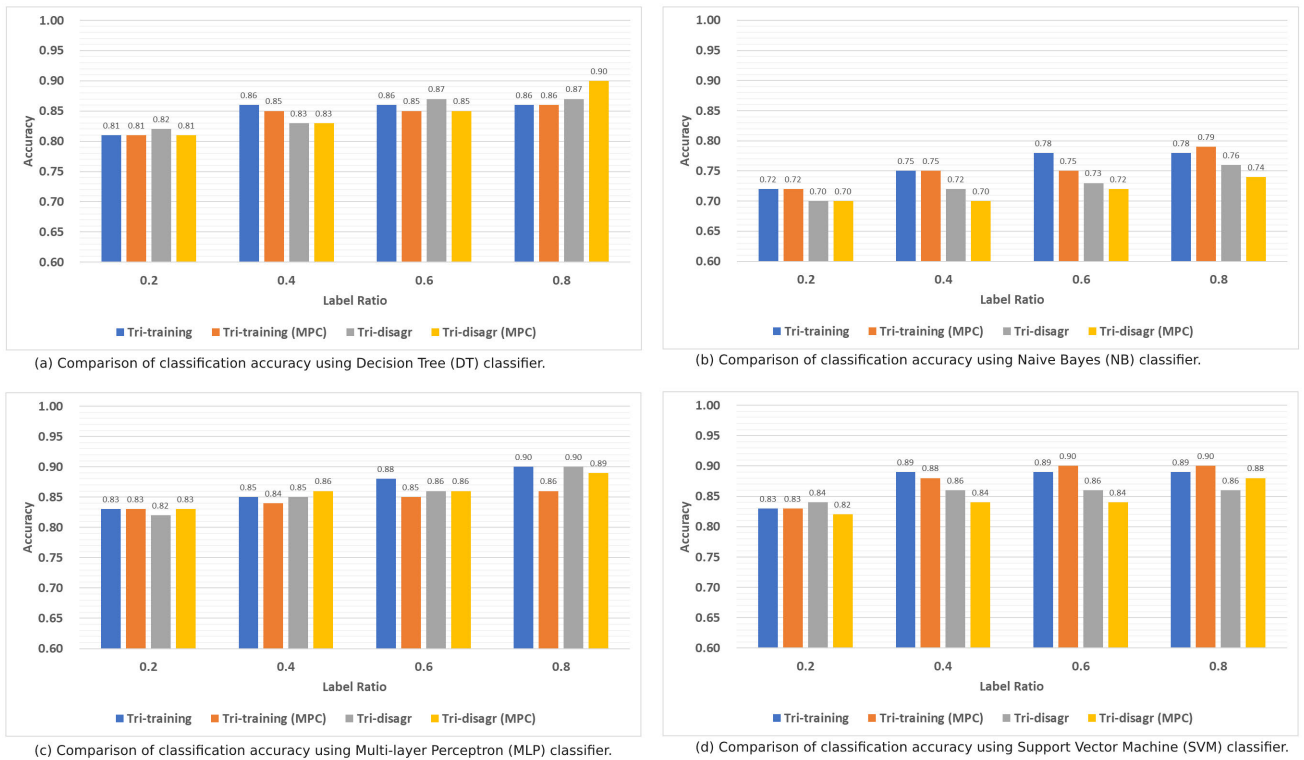


FIGURE 4. Comparison of classification accuracy of original and secure MPC of tri-training, tri-training with disagreement algorithm on AUSTRALIAN dataset.

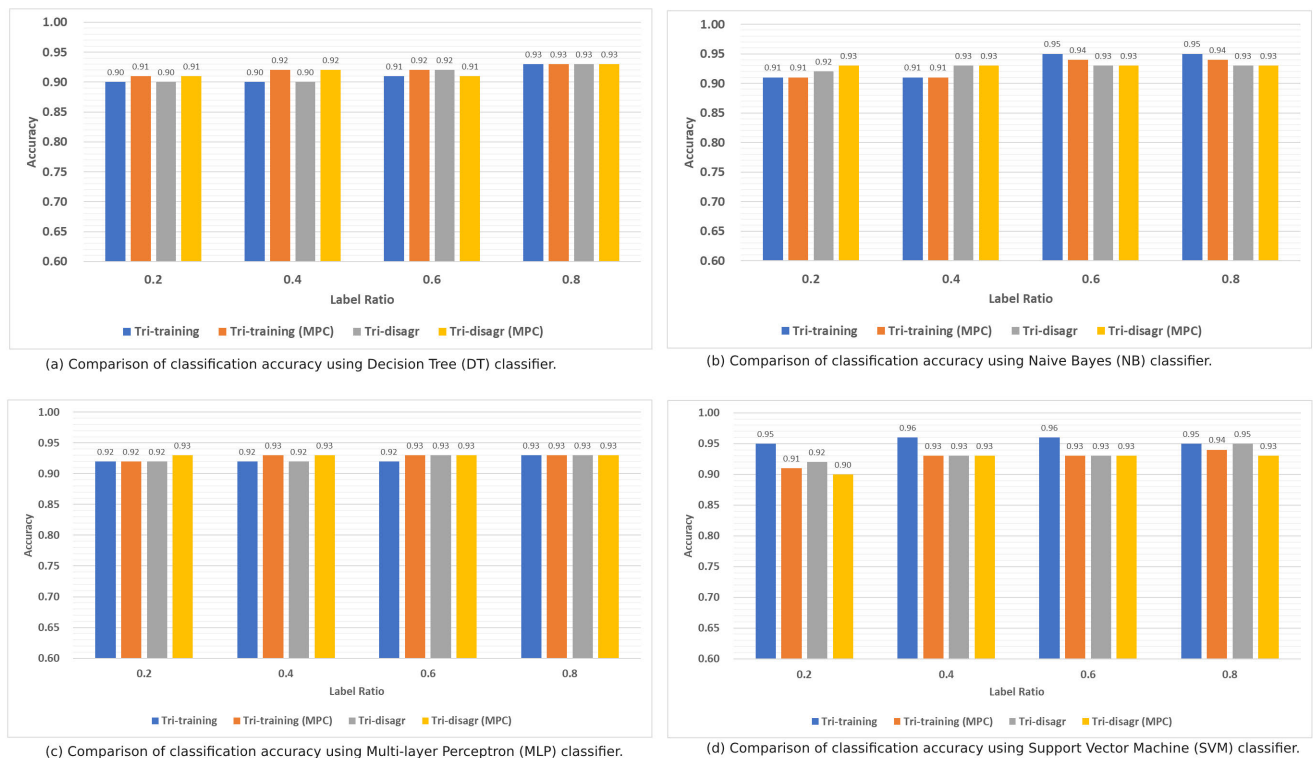


FIGURE 5. Comparison of classification accuracy of original and secure MPC of tri-training, tri-training with disagreement algorithm on WDBC dataset.

with disagreement in secure computation using artificial datasets. We use ten samples of artificial datasets with

varying noise intensity, dimensionality, extreme outliers, and non-uniform distribution aspect. For further evaluation,

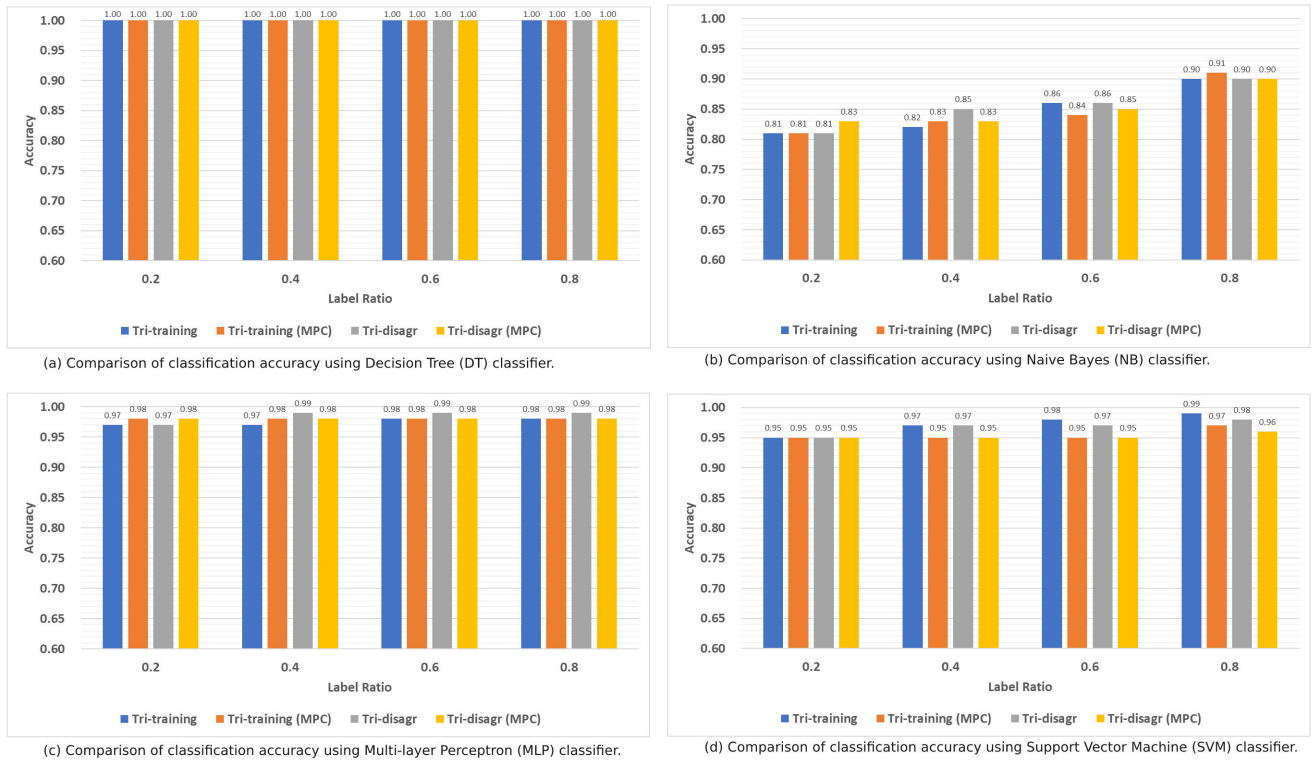


FIGURE 6. Comparison of classification accuracy of original and secure MPC of tri-training, tri-training with disagreement algorithm on MNIST dataset.

TABLE 4. Classification accuracy of original and secure MPC of tri-training, tri-training with disagreement algorithm on UNSW-NB15, HIGGS, and CIC-IDS dataset.

Classifier	Label Ratio	Tri-training	Tri-training (MPC)	Tri-disagr	Tri-disagr (MPC)
Decision Tree	0.2	1.00	1.00	1.00	1.00
	0.4	1.00	1.00	1.00	1.00
	0.6	1.00	1.00	1.00	1.00
	0.8	1.00	1.00	1.00	1.00
Naive Bayes	0.2	1.00	1.00	1.00	1.00
	0.4	1.00	1.00	1.00	1.00
	0.6	1.00	1.00	1.00	1.00
	0.8	1.00	1.00	1.00	1.00
MLP	0.2	1.00	1.00	1.00	1.00
	0.4	1.00	1.00	1.00	1.00
	0.6	1.00	1.00	1.00	1.00
	0.8	1.00	1.00	1.00	1.00
SVM	0.2	1.00	1.00	1.00	1.00
	0.4	1.00	1.00	1.00	1.00
	0.6	1.00	1.00	1.00	1.00
	0.8	1.00	1.00	1.00	1.00

we examine all the algorithms with the dataset using Monte Carlo Cross-validation. A random subset of the data is chosen as the validation set and the model is trained on the remaining data. The simulation is repeated a number of times, with a different random subset chosen as the validation set each time.

First, we evaluate the performance of the original tri-training and tri-training in secure MPC, tri-training with disagreement, and tri-training with disagreement in secure MPC with varying noise intensity. Table 5 shows the accuracy of each algorithm for MNIST dataset generated with Gaussian noise using different variance value. Gaussian noise is distributed randomly and uniformly over the entire image of MNIST dataset. As we can see for all classifiers, there is no significant difference in the accuracy of the original tri-training algorithm compared to the tri-training in secure MPC. The same pattern also occurs in tri-training with disagreement compared to tri-training with disagreement in secure MPC.

As the noise intensity of the MNIST dataset increased, the performance of all algorithms decreased due to the difficulty in distinguishing between the true class labels and the noisy labels.

In the second evaluation, we analyze the performance of the original tri-training and tri-training in secure MPC, as well as tri-training with disagreement and tri-training with disagreement in secure MPC with non-uniform distribution datasets. In the aspect of non-uniform distribution, Forest Cover Type dataset and the Epileptic Seizure Recognition dataset are two commonly used datasets in machine learning [47], [48]. Table 6 shows the accuracy of each algorithm for the Forest Cover Type dataset and the Epileptic Seizure Recognition dataset. For all classifiers, the performance of

TABLE 5. Classification accuracy of original and secure MPC of tri-training, tri-training with disagreement algorithm different classifier on noisy artificial dataset.

Dataset Name	Dataset Category	Dimension (No. Class)	Classifier	Accuracy Tri-training	Accuracy Tri-training (MPC)	Accuracy Tri-disagr	Accuracy Tri-disagr (MPC)
MNIST Gaussian Noise mean = 0.0, var = 0.05	Noise Dataset	60,000 x 784 (10)	Decision Tree	0.89	0.89	0.89	0.89
			Naive Bayes	0.75	0.76	0.77	0.77
			MLP	0.84	0.83	0.84	0.84
			SVM	0.79	0.80	0.79	0.79
MNIST Gaussian Noise mean = 0.0, var = 0.1	Noise Dataset	60,000 x 784 (10)	Decision Tree	0.85	0.85	0.85	0.85
			Naive Bayes	0.71	0.71	0.72	0.72
			MLP	0.80	0.79	0.81	0.80
			SVM	0.76	0.76	0.75	0.75
MNIST Gaussian Noise mean = 0.0, var = 0.5	Noise Dataset	60,000 x 784 (10)	Decision Tree	0.81	0.81	0.81	0.82
			Naive Bayes	0.64	0.65	0.66	0.66
			MLP	0.75	0.75	0.75	0.75
			SVM	0.70	0.71	0.70	0.71
MNIST Gaussian Noise mean = 0.0, var = 0.7	Noise Dataset	60,000 x 784 (10)	Decision Tree	0.75	0.75	0.75	0.75
			Naive Bayes	0.60	0.60	0.61	0.61
			MLP	0.70	0.70	0.71	0.71
			SVM	0.66	0.67	0.66	0.66

all the algorithms decreases as the class distribution of the dataset becomes more non-uniform because the classifier will have a harder time correctly classifying the minority class. As observed for all classifiers, the classification accuracy results showed comparable performance between the original tri-training algorithm and the tri-training in secure MPC. Comparing tri-training with disagreement to tri-training with disagreement in secure MPC, the same pattern is also observed.

Next, we compare the results of the original tri-training, tri-training in secure MPC, tri-training with disagreement, and tri-training with disagreement in secure MPC on artificial datasets with various dimensions. We observed how the performance of all the algorithms changes as the dimensionality of the dataset increases. Table 7 shows the accuracy of each classifier and algorithm using artificial datasets featuring a wide variety of dimensions. The classification accuracy of all the algorithms decreases as the dimensionality of the dataset increased due to the curse of dimensionality. The classification accuracy did not differ significantly between the original tri-training algorithm and its implementation in secure MPC. Similarly, there was no significant difference in classification accuracy between tri-training with disagreement and its implementation in secure MPC.

Finally, we test the performance of the original tri-training and tri-training in secure MPC, as well as tri-training with disagreement and tri-training with disagreement in secure MPC using artificial datasets with extreme outliers. We observed how the performance of all the algorithms changes as the number of extreme outliers increase in the dataset. Table 8 shows the accuracy of each classifier and algorithm on extreme outliers datasets. The performance of all the algorithms decreases as the number of extreme outliers in the dataset increases because the presence of extreme outliers can have a significant impact on the classifier's decision

boundary. The comparison between the original tri-training and tri-training in secure MPC for extreme outlier datasets did not reveal any significant differences. Similarly, no marked differences were observed between tri-training with disagreement and tri-training with disagreement in secure MPC.

3) TWO-GROUPS STATISTICAL TEST

To provide a comprehensive comparison of tri-training and tri-training in secure computation, tri-training with disagreement and tri-training with disagreement in secure computation we provide a statistical t-test [53] for AUSTRALIAN, WDBC and MNIST dataset.

A t-test is an inferential statistic used to determine whether there is a statistically significant difference between the means of two groups that may be related in certain characteristics. The t-test establishes the problem statement by assuming that the two means are equal, using a sample from each of the two sets. The null hypothesis is that the population means are identical. The null hypothesis is stated as follows: $H_0: \mu_1 - \mu_2 = 0$, and the alternatives hypothesis $H_1: \mu_1 - \mu_2 \neq 0$ is that the population means are not identical. μ_1 and μ_2 represent the value of means from the first and second sample, respectively.

The statistical t-test is conducted for 16 observations in each algorithm based on the classification accuracy value of AUSTRALIAN, WDBC and MNIST dataset in Figure 4, 5, and 6, respectively.

Table 9 provides the t-test result for AUSTRALIAN dataset. In the table 9(a) the mean and variance value of tri-training is 0.836250 and 0.002972 respectively. The tri-training in secure MPC has a mean and variance 0.829375 and 0.002820, respectively. We accept the null hypothesis H_0 because the p -value (0.720358) is bigger than the level of significance (Alpha = 0.05). The mean and variance of original tri-training with disagreement are 0.821875 and

TABLE 6. Classification accuracy of original and secure MPC of tri-training, tri-training with disagreement algorithm different classifier on non-uniform distribution artificial dataset.

Dataset Name	Dataset Category	Dimension (No. Class)	Classifier	Accuracy Tri-training	Accuracy Tri-training (MPC)	Accuracy Tri-disagr	Accuracy Tri-disagr (MPC)
Forest Cover Type	Non-uniform Distribution	581,012 x 55 (7)	Decision Tree	0.89	0.90	0.88	0.88
			Naive Bayes	0.65	0.66	0.65	0.66
			MLP	0.68	0.69	0.69	0.69
			SVM	0.63	0.63	0.64	0.64
Epileptic Seizure Recognition	Non-uniform Distribution	11,500 x 178 (5)	Decision Tree	0.73	0.73	0.72	0.72
			Naive Bayes	0.74	0.74	0.74	0.74
			MLP	0.59	0.59	0.60	0.61
			SVM	0.59	0.60	0.60	0.60

TABLE 7. Classification accuracy of original and secure MPC of tri-training, tri-training with disagreement algorithm different classifier on multi-dimension artificial dataset.

Dataset Name	Dataset Category	Dimension (No. Class)	Classifier	Accuracy Tri-training	Accuracy Tri-training (MPC)	Accuracy Tri-disagr	Accuracy Tri-disagr (MPC)
DS Dimension 1	Dimensionality	10,000 x 50 (2)	Decision Tree	0.89	0.90	0.89	0.89
			Naive Bayes	0.85	0.86	0.85	0.85
			MLP	0.84	0.84	0.83	0.84
			SVM	0.86	0.86	0.86	0.87
DS Dimension 2	Dimensionality	10,000 x 100 (2)	Decision Tree	0.86	0.87	0.86	0.87
			Naive Bayes	0.83	0.83	0.83	0.84
			MLP	0.82	0.83	0.82	0.82
			SVM	0.83	0.83	0.84	0.84
DS Dimension 3	Dimensionality	10,000 x 150 (2)	Decision Tree	0.82	0.83	0.82	0.82
			Naive Bayes	0.82	0.81	0.81	0.81
			MLP	0.79	0.80	0.81	0.81
			SVM	0.80	0.80	0.80	0.80
DS Dimension 4	Dimensionality	10,000 x 200 (2)	Decision Tree	0.76	0.76	0.77	0.77
			Naive Bayes	0.76	0.75	0.75	0.75
			MLP	0.70	0.70	0.73	0.73
			SVM	0.71	0.70	0.72	0.72

TABLE 8. Classification accuracy of original and secure MPC of tri-training, tri-training with disagreement algorithm different classifier on extreme outlier artificial dataset.

Dataset Name	Dataset Category	Dimension (No. Class)	Classifier	Accuracy Tri-training	Accuracy Tri-training (MPC)	Accuracy Tri-disagr	Accuracy Tri-disagr (MPC)
DS Outlier 1	Number of Extreme Outlier = 2	1,000 x 5 (2)	Decision Tree	0.84	0.84	0.85	0.85
			Naive Bayes	0.82	0.82	0.83	0.83
			MLP	0.83	0.83	0.86	0.86
			SVM	0.80	0.79	0.78	0.78
DS Outlier 2	Number of Extreme Outlier = 11	1,000 x 5 (2)	Decision Tree	0.81	0.81	0.82	0.83
			Naive Bayes	0.80	0.79	0.80	0.80
			MLP	0.75	0.76	0.75	0.76
			SVM	0.76	0.76	0.75	0.76
DS Outlier 3	Number of Extreme Outlier = 24	1,000 x 5 (2)	Decision Tree	0.76	0.76	0.75	0.76
			Naive Bayes	0.77	0.77	0.78	0.79
			MLP	0.72	0.71	0.72	0.72
			SVM	0.69	0.69	0.70	0.71

0.003683, respectively, while the mean and variance of tri-training with disagreement in secure MPC are 0.816875 and 0.004343, respectively as presented in table 9(b). The p -value is 0.824856, we accept null hypothesis H_0 which indicates the distribution of classification accuracy for original tri-training

with disagreement is equal to tri-training with disagreement in secure MPC.

The results of the t-test for the WDBC dataset are presented in Table 10. The mean and variance value of tri-training are shown to be 0.929375 and 0.000446 in table 10(a), respec-

TABLE 9. Comparison of Tri-training with disagreement and Tri-training with disagreement (MPC).

(a) Comparison of Tri-training and Tri-training (MPC)

Algorithm	Tri-training	Tri-training (MPC)
Mean	0.836250	0.829375
Variance	0.002972	0.002820
Observations	16	16
<i>P</i> -value	0.720358	

(b) Comparison of Tri-training with disagreement and Tri-training with disagreement (MPC)

Algorithm	Tri-disagr	Tri-disagr (MPC)
Mean	0.821875	0.816875
Variance	0.003683	0.004343
Observations	16	16
<i>P</i> -value	0.824856	

TABLE 10. Comparison of Tri-training with disagreement and Tri-training with disagreement (MPC).

(a) Comparison of Tri-training and Tri-training (MPC)

Algorithm	Tri-training	Tri-training (MPC)
Mean	0.929375	0.925000
Variance	0.000446	0.000120
Observations	16	16
<i>P</i> -value	0.467796	

(b) Comparison of Tri-training with disagreement and Tri-training with disagreement (MPC)

Algorithm	Tri-disagr	Tri-disagr (MPC)
Mean	0.924375	0.925100
Variance	0.000146	0.000093
Observations	16	16
<i>P</i> -value	0.872771	

tively. The mean and variance of the tri-training in secure MPC are 0.925000 and 0.000120, respectively. We are willing to concede that the null hypothesis H_0 is correct because the *p*-value (0.467796) is greater than the level of significance (Alpha = 0.05). According to the data presented in table 10(b), the mean and variance of original tri-training with disagreement are 0.924375 and 0.000146, respectively, whereas the mean and variance of tri-training with disagreement in secure MPC are 0.925100 and 0.000093, respectively. We are able to accept the null hypothesis because the *p*-value is 0.872771. H_0 shows that the distribution of classification accuracy for the original tri-training with disagreement is the same as the distribution of classification accuracy for tri-training with disagreement in secure MPC.

Table 11 shows the result of the t-test for the MNIST dataset. In table 11(a), the mean value of tri-training is 0.948750 and the variance is 0.004172. Tri-training in secure MPC has a mean of 0.945625 and a variance of 0.004093. We accept the null hypothesis H_0 because the *p*-value (0.891555) is bigger than the level of significance (Alpha = 0.05). The mean and variance for original tri-training with

TABLE 11. Comparison of Tri-training with disagreement and Tri-training with disagreement (MPC).

(a) Comparison of Tri-training and Tri-training (MPC)

Algorithm	Tri-training	Tri-training (MPC)
Mean	0.948750	0.945625
Variance	0.004172	0.004093
Observations	16	16
<i>P</i> -value	0.891555	

(b) Comparison of Tri-training with disagreement and Tri-training with disagreement (MPC)

Algorithm	Tri-disagr	Tri-disagr (MPC)
Mean	0.951875	0.946250
Variance	0.003803	0.003652
Observations	16	16
<i>P</i> -value	0.796183	

TABLE 12. Statistical t-test result with Alpha = 0.05 for comparison of the original tri-training and tri-training (MPC), tri-training with disagreement and tri-training with disagreement (MPC) on artificial dataset.

Classifier	Compared Algorithm	<i>P</i> -value	H_0 Result
Decision Tree	Tri-training vs Tri-training (MPC)	0.03952	Accept
	Tri-disagr vs Tri-disagr (MPC)	0.01798	Accept
Naive Bayes	Tri-training vs Tri-training (MPC)	0.72136	Accept
	Tri-disagr vs Tri-disagr (MPC)	0.08210	Accept
MLP	Tri-training vs Tri-training (MPC)	0.72136	Accept
	Tri-disagr vs Tri-disagr (MPC)	0.33705	Accept
SVM	Tri-training vs Tri-training (MPC)	0.67269	Accept
	Tri-disagr vs Tri-disagr (MPC)	0.19015	Accept

disagreement are 0.951875 and 0.003803, respectively. For tri-training with disagreement in secure MPC, the mean and variance are 0.946250 and 0.003652, as shown in table 11(b). The *p*-value is 0.796183, so we accept the null hypothesis. The t-test result means that the distribution of classification accuracy for original tri-training with disagreement is the same as for tri-training with disagreement in secure MPC.

To prove that the original tri-training algorithm has similarities with tri-training in secure MPC, as well as tri-training with disagreement and tri-training with disagreement in secure MPC on artificial datasets. We compare tri-training and tri-training (MPC), tri-training with disagreement and tri-training with disagreement (MPC) using statistical test. Each test compares two groups of population with 10 samples. Rodriguez-Fdez et al., [54] provide a web platform for the comparison of algorithms using statistical tests. We provide a preliminary normality and homoscedasticity test on pairing between two groups. The normality of each group was tested using a Shapiro-Wilks test with alpha 0.1 and the homoscedasticity between groups was tested using Levene test with alpha 0.1. We use parametric t-test paired samples based on the normality and homoscedasticity results. Table 12 shows statistical t-test results with alpha = 0.05 for comparison of the original tri-training and tri-training

TABLE 13. Execution time (in second) for every dataset with different classifier and label ratio = 0.2.

Dataset	Classifier	Time Tri-training	Time Tri-training (MPC)	Time Tri-disagr	Time Tri-disagr (MPC)	Compare Time Tri-training (MPC) / Tri-training	Compare Time Tri-disagr (MPC) / Tri-disagr
AUSTRALIAN	Decision Tree	0.03	0.28	0.03	0.17	9	6
	Naive Bayes	0.57	0.78	0.18	0.65	1	4
	MLP	0.03	0.81	0.02	0.69	27	35
	SVM	0.10	0.89	0.06	0.75	9	13
WDBC	Decision Tree	0.04	0.28	0.03	0.20	7	7
	Naive Bayes	0.31	0.73	0.77	0.85	2	1
	MLP	0.04	0.89	0.03	0.77	22	26
	SVM	0.08	0.81	0.06	0.93	10	16
MNIST	Decision Tree	29.15	36.55	16.98	35.04	1	2
	Naive Bayes	13.75	58.33	13.49	83.55	4	6
	MLP	86.41	113.01	112.44	138.42	1	1
	SVM	754.07	1,016.52	497.17	796.87	1	2
UNSW-NB15	Decision Tree	174.33	667.70	122.05	327.16	4	3
	Naive Bayes	340.04	473.52	224.08	265.84	1	1
	MLP	155.10	176.17	140.61	157.41	1	1
	SVM	1,531.30	1,989.87	974.13	1,077.67	1	1
HIGGS	Decision Tree	442.47	1,494.66	309.77	1,659.54	3	5
	Naive Bayes	847.99	1,880.85	558.79	1,177.50	2	2
	MLP	383.92	2,379.13	348.04	2,082.86	6	6
	SVM	3,906.37	7,831.30	2,485.01	3,836.91	2	2
CIC-IDS	Decision Tree	1,769.88	6,690.14	1,139.08	4,970.31	4	4
	Naive Bayes	3,358.05	6,701.26	2,012.82	5,577.44	2	3
	MLP	7,520.31	17,012.28	7,178.25	15,044.45	2	2
	SVM	14,375.45	63,539.47	9,044.85	50,293.52	4	6

(MPC), tri-training with disagreement and tri-training with disagreement. The null hypothesis H_0 indicates two related or repeated samples have identical mean values.

4) EXECUTION TIME COMPARISON

As shown in subsection V-B1, there is no significant difference in classification accuracy between original tri-training and tri-training in secure MPC, and either original tri-training with disagreement compared to tri-training with disagreement in secure MPC for different values of *label ratio*. In this subsection, we will discuss the execution time for each dataset with different classifiers using the same value of *label ratio* = 0.2. Considering *label ratio* = 0.2 reflects a realistic setting with a large dataset, where labeled data is limited compared to a massive amount of unlabeled data. Moreover, another *label ratio* has the same tendency to *label ratio* = 0.2.

$$CompTimeTriTrain = \frac{TimeTriTrainMPC}{TimeTriTrain} \tag{5}$$

$$CompTimeTriDisagr = \frac{TimeTriDisagrMPC}{TimeTriDisagr} \tag{6}$$

We provide definition for execution time comparison *CompTimeTriTrain* in equation (5) as comparison of execution time between tri-training in secure MPC and original tri-training algorithm, and *CompTimeTriDisagr* in equation (6) as comparison of execution time between tri-training with disagreement in secure MPC and original tri-training with disagreement algorithm.

Table 13 shows the execution time for every dataset with different classifier and the value of *label ratio* = 0.2. For AUSTRALIAN dataset, decision tree classifier has fastest execution time compared to the other classifiers. The execution time of tri-training in secure MPC is 9 times longer than original tri-training algorithm. Tri-training with disagreement in secure MPC takes 6 times longer to execute than the original tri-training with disagreement algorithm. Naive Bayes classifier has the lowest increasing execution time while comparing tri-training in secure MPC to original tri-training algorithm. On average, for all classifiers in AUSTRALIAN dataset tri-training in secure MPC takes 12 times longer than original tri-training algorithm, while the execution time is 14 times longer for tri-tri-training with disagreement in secure MPC compared to original tri-training with disagreement. In the WDBC dataset, the average value of *CompTimeTriTrain* and *CompTimeTriDisagr* is 10 and 12 respectively. The average *CompTimeTriTrain* value is 2 and the average *CompTimeTriDisagr* value is 3 for MNIST dataset. For large datasets like UNSW-NB15, HIGGS, and CIC-IDS the average values of *CompTimeTriTrain* and *CompTimeTriDisagr* become smaller. UNSW-NB15 dataset has the same average value of *CompTimeTriTrain* and *CompTimeTriDisagr*, equal to 2. Both HIGGS and CIC-IDS dataset has the average value of *CompTimeTriTrain* and *CompTimeTriDisagr* 3 and 4 respectively.

In all datasets we can see that tri-training in secure MPC and tri-training with disagreement in secure MPC take longer

execution time compared to original tri-training and tri-training with disagreement algorithm, due to the fact that privacy-preserving secure MPC requires higher computational time with three parties. With the increasing size of the dataset, the average value of *CompTimeTriTrain* and *CompTimeTriDisagr* decreased.

VI. CONCLUSION

In this paper, we present and analyze the privacy-preserving distributed data mining for semi-supervised tri-training classification using secure multi-party computation. We have evaluated our proposed application extensively using four learning algorithms and six well-known datasets that vary from small to extremely large dimensions. Based on the simulation, the proposed application of tri-training in secure MPC and tri-training with disagreement in secure MPC has the same classification accuracy compared to the original tri-training and tri-training with disagreement algorithm. The result also shows a secure-MPC scheme can preserve privacy in tri-training and tri-training with disagreement algorithm while keeping accuracy. Moreover, we have presented the reliability of the proposed application by calculating the average comparison value of execution time for each. With the increasing dataset size, the difference of execution time decreased. This circumstance is reliable to the real-world application which consists of a massive amount of data. In future work, we plan to explore some possibilities to lower the execution time by analyzing the unlabeled data, and extending the proposed application to follow the malicious adversaries' security, which is beyond the scope of this work.

REFERENCES

- [1] K. D. Strang and Z. Sun, "Hidden big data analytics issues in the healthcare industry," *Health Informat. J.*, vol. 26, no. 2, pp. 981–998, Jun. 2020.
- [2] N. Domadiya and U. P. Rao, "Privacy preserving distributed association rule mining approach on vertically partitioned healthcare data," *Proc. Comput. Sci.*, vol. 148, pp. 303–312, Jan. 2019.
- [3] M. T. Pandian, P. Damodharan, S. Singh, and A. K. Aggarwal, "Secured medical document extraction in e-health care system using data mining techniques," *Mater. Today, Proc.*, to be published.
- [4] F. Zerka, S. Barakat, S. Walsh, M. Bogowicz, R. T. H. Leijenaar, A. Jochems, B. Miraglio, D. Townend, and P. Lambin, "Systematic review of privacy-preserving distributed machine learning from federated databases in health care," *JCO Clin. Cancer Informat.*, no. 4, pp. 184–200, Nov. 2020.
- [5] A. Aminifar, M. Shokri, F. Rabbi, V. K. I. Pun, and Y. Lamo, "Extremely randomized trees with privacy preservation for distributed structured health data," *IEEE Access*, vol. 10, pp. 6010–6027, 2022.
- [6] J. Liu, Y. Tian, Y. Zhou, Y. Xiao, and N. Ansari, "Privacy preserving distributed data mining based on secure multi-party computation," *Comput. Commun.*, vol. 153, pp. 208–216, Mar. 2020.
- [7] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, and Y.-A. Tan, "Secure multi-party computation: Theory, practice and applications," *Inf. Sci.*, vol. 476, pp. 357–372, Feb. 2019.
- [8] V. Baby and N. Subhash, "Privacy-preserving distributed data mining techniques: A survey," *Int. J. Comput. Appl.*, vol. 143, no. 10, pp. 37–41, Jun. 2016.
- [9] C. Sun, L. Ippel, A. Dekker, M. Dumontier, and J. Van Soest, "A systematic review on privacy-preserving distributed data mining," *Data Sci.*, vol. 4, no. 2, pp. 121–150, 2021.
- [10] K. Liu, H. Kargupta, and J. Ryan, "Random projection-based multiplicative data perturbation for privacy preserving distributed data mining," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 1, pp. 92–106, Jan. 2006.
- [11] M. A. P. Chamikara, P. Bertok, S. Camtepe, I. Khalil, and D. Liu, "Efficient data perturbation for privacy preserving and accurate data stream mining," *Pervasive Mobile Comput.*, vol. 48, pp. 1–19, Aug. 2018.
- [12] A.-T. Tran, T.-D. Luong, J. Karnjana, and V.-N. Huynh, "An efficient approach for privacy preserving decentralized deep learning models based on secure multi-party computation," *Neurocomputing*, vol. 422, pp. 245–262, Jan. 2021.
- [13] T. Araki, A. Barak, J. Furukawa, M. Keller, Y. Lindell, K. Ohara, and H. Tsuchida, "Generalizing the SPDZ compiler for other protocols," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 880–895.
- [14] D. Bogdanov, M. Niitsoo, T. Toft, and J. Willemsen, "High-performance secure multi-party computation for data mining applications," *Int. J. Inf. Secur.*, vol. 11, no. 6, pp. 403–418, Nov. 2012.
- [15] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1175–1191.
- [16] E. Sotthiwat, L. Zhen, Z. Li, and C. Zhang, "Partially encrypted multi-party computation for federated learning," in *Proc. IEEE/ACM 21st Int. Symp. Cluster, Cloud Internet Comput. (CCGrid)*, May 2021, pp. 828–835.
- [17] J. Zhao, N. Liu, and A. Malov, "Safe semi-supervised classification algorithm combined with active learning sampling strategy," *J. Intell. Fuzzy Syst.*, vol. 35, no. 4, pp. 4001–4010, 2018.
- [18] R. Fierimonte, S. Scardapane, A. Uncini, and M. Panella, "Fully decentralized semi-supervised learning via privacy-preserving matrix completion," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 11, pp. 2699–2711, Nov. 2016.
- [19] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach. Learn.*, vol. 109, no. 2, pp. 373–440, 2020.
- [20] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proc. 11th Annu. Conf. Comput. Learn. Theory*, Jul. 1998, pp. 92–100.
- [21] S. Goldman and Y. Zhou, "Enhancing supervised learning with unlabeled data," in *Proc. ICML*. Princeton, NJ, USA: Citeseer, 2000, pp. 327–334.
- [22] Z.-H. Zhou and M. Li, "Tri-training: Exploiting unlabeled data using three classifiers," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 11, pp. 1529–1541, Nov. 2005.
- [23] Q. Zhao, C. Zhao, S. Cui, S. Jing, and Z. Chen, "PrivateDL: Privacy-preserving collaborative deep learning against leakage from gradient sharing," *Int. J. Intell. Syst.*, vol. 35, no. 8, pp. 1262–1279, Aug. 2020.
- [24] G. Han, T. Zhang, Y. Zhang, G. Xu, J. Sun, and J. Cao, "Verifiable and privacy preserving federated learning without fully trusted centers," *J. Ambient Intell. Humanized Comput.*, vol. 13, no. 3, pp. 1431–1441, Mar. 2022.
- [25] Z. Li and Z. Li, "Inductive and effective privacy-preserving semi-supervised learning with harmonic anchor mixture," in *Proc. Int. Symp. Electr. Electron. Inf. Eng.*, Feb. 2021, pp. 507–516.
- [26] D. Demmler, T. Schneider, and M. Zohner, "ABY—A framework for efficient mixed-protocol secure two-party computation," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2015, pp. 1–15.
- [27] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Found. Comput. Sci. (SFCS)*, Nov. 1982, pp. 160–164.
- [28] O. Goldreich, "Secure multi-party computation," *Manuscript. Preliminary Version*, vol. 78, p. 110, Jun. 1998.
- [29] M. Keller, "MP-SPDZ: A versatile framework for multi-party computation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 1575–1590.
- [30] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, "CrypTen: Secure multi-party computation meets machine learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021.
- [31] B. M. Shahshahani and D. A. Landgrebe, "The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon," *IEEE Trans. Geosci. Remote Sens.*, vol. 32, no. 5, pp. 1087–1095, Sep. 1994.
- [32] D. J. Miller and H. Uyar, "A mixture of experts classifier with learning based on both labelled and unlabelled data," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 9, 1996.
- [33] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Mach. Learn.*, vol. 39, nos. 2–3, pp. 103–134, 2000.

- [34] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. 16th Int. Conf. Mach. Learn.*, vol. 99, 1999, pp. 200–209.
- [35] B. Jiang, H. Chen, B. Yuan, and X. Yao, "Scalable graph-based semi-supervised learning through sparse Bayesian model," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 12, pp. 2758–2771, Dec. 2017.
- [36] D. Zhou, T. Hofmann, and B. Schölkopf, "Semi-supervised learning on directed graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 17, 2004.
- [37] H. He, D. Han, and J. Dezert, "Disagreement based semi-supervised learning approaches with belief functions," *Knowl.-Based Syst.*, vol. 193, Apr. 2020, Art. no. 105426.
- [38] A. Søgaard, "Simple semi-supervised training of part-of-speech taggers," in *Proc. ACL Conf. Short Papers*, 2010, pp. 205–208.
- [39] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Prague, Czech Republic: Springer, May 1999, pp. 223–238.
- [40] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-throughput semi-honest secure three-party computation with an honest majority," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 805–817.
- [41] W. N. Street, W. H. Wolberg, and O. L. Mangasarian, "Nuclear feature extraction for breast tumor diagnosis," *Proc. SPIE*, vol. 1905, Jul. 1993, pp. 861–870.
- [42] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.
- [43] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
- [44] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning," *Nature Commun.*, vol. 5, no. 1, pp. 1–9, Jul. 2014.
- [45] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, vol. 1, Jan. 2018, pp. 108–116.
- [46] T. S. Nazaré, G. B. P. da Costa, W. A. Contato, and M. Ponti, "Deep convolutional neural networks and noisy images," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Valparaíso, Chile: Springer, Nov. 2017, pp. 416–424.
- [47] S. O. Sahin and S. S. Kozat, "Nonuniformly sampled data processing using LSTM networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 5, pp. 1452–1461, May 2019.
- [48] C. Guo, W. Liu, X. Liu, and Y. Zhang, "Secure similarity search over encrypted non-uniform datasets," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 2102–2117, Jul. 2022.
- [49] P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to Data Mining*, 2nd ed. London, U.K.: Pearson, 2018.
- [50] H. Zhang, "The optimality of naive Bayes," *Aa*, vol. 1, no. 2, p. 3, 2004.
- [51] R. Collobert and S. Bengio, "Links between perceptrons, MLPs and SVMs," in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*, 2004, p. 23.
- [52] N. Li, F. He, W. Ma, R. Wang, and X. Zhang, "Wind power prediction of kernel extreme learning machine based on differential evolution algorithm and cross validation algorithm," *IEEE Access*, vol. 8, pp. 68874–68882, 2020.
- [53] E. H. Livingston, "Who was student and why do we care so much about his *t*-test?," *J. Surgical Res.*, vol. 118, no. 1, pp. 58–65, May 2004.
- [54] I. Rodriguez-Fdez, A. Canosa, M. Mucientes, and A. Bugarin, "STAC: A web platform for the comparison of algorithms using statistical tests," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, Aug. 2015, pp. 1–8.



HENDRA KURNIAWAN (Member, IEEE) received the B.S. degree in informatics engineering from Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia, in 2007, the M.Sc.Eng. degree in computer science from the National Taiwan University of Science and Technology, Taiwan, in 2010, and the Ph.D. degree in computer science from Kanazawa University, Japan, in 2023. His research interests include privacy-preserving data mining, privacy issue in machine learning, and security and privacy issue for edge AI.



MASASHIRO MAMBO (Member, IEEE) received the B.Eng. degree from Kanazawa University, Kanazawa, Japan, in 1988, and the M.S.Eng. and Dr.Eng. degrees in electronic engineering from the Tokyo Institute of Technology, Tokyo, Japan, in 1990 and 1993, respectively. In 2011, after working with the Japan Advanced Institute of Science and Technology, Tohoku University, Sendai, Japan, and the University of Tsukuba, Tsukuba, Japan, he joined Kanazawa University. He is currently a Professor with the Faculty of Electrical, Information and Communication Engineering, Institute of Science and Engineering. His research interests include information security, software protection, and privacy protection. He served as the Co-Editor-in-Chief for the *International Journal of Information Security*, the Steering Committee Chair for the International Conference on Information Security, and the Chair for the Technical Committee on Information Security in Engineering Sciences Society of the Institute of Electronics, Information and Communication Engineers (ISEC in ESS of IEICE).