Blind Inference: An Automated Privacy-Preserving Prediction Service using Secure Multi-Party Computation for Medical Applications

Gharib Gharibi, PhD[1], Babak P. Gilkalaye, MS[1], Praneeth Vepakomma, PhD[2], I. Zachi Attia, PhD[3], Ravi Patel, BS[1], David Wagner, MTS[1], Andrew Rademacher, BS[1], Riddhiman Das, MS[1], Ramesh Raskar, PhD[2], Paul A. Friedman, MD[3], Suraj Kapa, MD[1]

[1] TripleBlind, Inc., Kansas, City, MO, USA;
[2] MIT Media Lab, Massachusetts Institute of Technology, Cambridge, MA, USA;
[3] Department of Cardiovascular Medicine, Mayo Clinic, Rochester, MN, USA

Abstract
Machine-learning prediction services play significant role in advancing the medical applications, such as DAN sequence alignment, patient diagnosis, and model evaluation. Existing prediction services either require installing the model on the user's device or uploading the data to the service provider. The former approach hampers the model's intellectual property while the latter is extremely challeging due to privacy, ethical, and legal concerns. Secure Multi-Party Computation (MPC) enables joint computations on data owned by individual parties without revealing their data to each other, making it suitable for privacy-preserving predictions. However, most of the existing MPC methods require knowledge about the underlying cryptography primitives, which puts them out of reach from non-cryptography experts; e.g., machine learning practitioners. To this end, we present Blind Inference*, an automated end-to-end system for secure MPC prediction services. We focus in this paper on our ML-first API that facilitates privacy-preserving inference for non-cryptography experts. Specifically, we shed light on our innovation MPC approach, explain our system architecture and implementation, and demonstrate through our thorough evaluation that our underlying algorithm and toolset is at least* $3.5\times$ *faster compared to the current state-of-the-art tools while preserving* 100 *% accuracy.*

1 INTRODUCTION

Recent advances in machine learning (ML) methods and tools have led to significant achievements across an ever-growing number of domains, especially in the medical field[1,2]. Reliable ML models, often used as model-as-a-service (aka., inference-as-a-service), represent great potential for collaboration, research, and commercialization[3]. However, the majority of existing prediction services either require installing the model on the user's device or require the user to upload their test data to the provider's service. The former approach is challenging due to competition and intellectual property concerns: models are core to the organization's business and are expensive to train. Alternatively, requiring the user to upload their data to the service provider is even more challenging, especially in the medical domain, due to privacy concerns and regulations. Therefore, centralized predictions systems are undesired due to competition, privacy, and legal concerns[4–7].

To mitigate the aforementioned privacy risks, a number of studies suggested the usage of cryptography-based techniques, including secure multi-party computation (MPC)[8], fully homomorphic encryption (FHE)[9], and oblivious transfer[10]. However, the significant computational overhead required by FHE and oblivious transfer makes them not suitable for real-world applications. For example, running a secure inference on a simple `LeNet-5`[11] model with Maxpool layers and a single MNIST[12] sample using HE-Transformer[13], a homomorphic encryption library, takes over 548 seconds to complete and results in over 4% accuracy loss. In contrast, our work, an MPC-based method and toolset, can run the same inference (prediction) in less than 170 milliseconds without any loss in accuracy .

Secure multi-party computation (MPC) is a cryptographic protocol that enables individual parties to perform joint computations (e.g., model inference) using their decentralized data and models without revealing any of their private inputs, i.e., test data (owned by the organization requesting the service) and model (owned by the organization providing the service). Generally, such systems start by encrypting both the model and the data and then utilize different cryptography techniques, e.g., *additive secret sharing*, to calculate the inference result over the encrypted inputs and finally reveal the inference results to the service requester.

While MPC-based inference systems for neural networks have illustrated some promising results[14–17], such works, besides *CrypTen*[14], largely ignore the design and implementation of practical tool support, making them hard to adopt in real-world applications. Moreover, majority of the existing work still require intensive computations and communications between the involved parties–leading to impractical execution times, especially for medical applications.

To this end, we demonstrate an innovative secure MPC method and a toolset, called *Blind Inference*, focused towards secure and privacy-preserving inference (prediction) for neural networks over the public Internet. The contributions of this paper are summarized as follows:

- We present the design and implementation of a complete system for privacy-preserving prediction service. Our goal here is to facilitate and accelerate the adoption of MPC-based methods and tools by non-cryptography experts, especially in the medical domain, which in return can aid multi-institutional and multi-national medical applications while preserving the model and data privacy.
- We conduct thorough experiments to evaluate the efficiency and accuracy of Blind Inference compared to prior work and then demonstrate its usability given a realistic medical application.

Our evaluation considers three types of experiments: Experiment A compares our MPC algorithm to prior algorithms in this domain over three types of models. Experiment B investigates the precision of our prediction results using a real-world use case conducted mainly by physician, ML practitioners. This experiment illustrates that our algorithm produces prediction results accurate up to the fourth decimal point. Experiment C compares the overall running time of a prediction task over three networks using most popular MPC tools, CrypTen and PySyft. This experiment demonstrates that our toolset is at least $3.5\times$ faster than these tools. Moreover, our tool provides a complete solution and an automated functionality accessible to ML practitioners without requiring MPC experience.

2 BACKGROUND

Secure MPC allows individual parties to jointly perform a computation on their private data without revealing it to each other. The idea of secure computation was first introduced by Yao et al.[8], who later proposed the *Garbled Circuits* approach, which then along with *Shamir Secret Sharing*[18] became the foundation of most secure MPC protocols.

Formally, in an MPC setup there exists n number of parties, $P_1, ..., P_n$, each holding some input data, $x_1, ..., x_n$ such that x_1 is held by party P_1, and wanting to compute a function f on their combined data, $f(x_1, ..., x_n)$, in a secure way such that no party learns others' data. For example, one party could hold a trained model and another party holds some data for inference. Using secure MPC, both parties can compute the inference results without revealing neither the data nor the model to each other. The goal of secure MPC protocol is twofold: (1) to achieve the computation f such that its output is equivalent to performing it on a centralized trusted party and (2) the computation must be protected even if some party acts in malicious behavior.

There are two main types of MPC implementations: Arithmetic circuits[19] and Boolean circuits[20]. Arithmetic circuits are functions consisting of addition and multiplication operations (also called gates). Polynomials are a good example of arithmetic circuits, but square roots and natural logarithms are not considered arithmetic circuits, for example. Addition gates can be computed at each party without communication, while multiplication gates often require several rounds of communications. Boolean circuits are a special case of the Arithmetic circuits operating in the Binary field, consisting of `XOR` and `AND` gates only. `XOR` is faster to compute than `AND` gates, which are much slower. Our approach uses a combination of both arithmetic and binary circuits and a set of protocols to convert between the two in order to compute the prediction results in a secure manner.

2.1 Secure MPC Addition Example

The main idea of secure MPC is that each party needs to create shares of its own data, exchange these shares with other parties, compute the results of the shares, and return the final output. To simplify understanding the secure MPC protocol, we illustrate how three parties, each with a private number, can compute the sum of their numbers without revealing them to each other. Figure 1 illustrates this example. Assume three parties, $A, B,$ and C, each holding a private number and want to compute the sum of their numbers without revealing them to each other. Assume party A holds the number $x_1 = 10$ and parties B and C hold the numbers $x_2 = -4$ and $x_3 = 15$, respectively. We want to compute the result of the function $f(x_1, x_2, x_3) = x_1 + x_2 + x_3$ securely. The sum of these three numbers is 21, but we cannot add them directly because none of the parties is welling to share their data in plain text with the other parties. To compute this addition securely using MPC, the parties follow these steps:

- First, each party creates three shares of its data:

  Party 1 creates the following shares:
  $x_{11} = 4, x_{12} = 4, x_{13} = 2$, thus $x_{11} + x_{12} + x_{13} = x_1$

  Party 2 creates the following shares:
  $x_{21} = 10, x_{22} = -7, x_{23} = -7$, thus $x_{21} + x_{22} + x_{23} = x_2$

  Party 3 creates the following shares:
  $x_{31} = 20, x_{32} = -4, x_{33} = -1$, thus $x_{31} + x_{32} + x_{33} = x_3$

- Second, each party sends two shares to the other parties: (i.e., x_{i1} to party 1, x_{i2} to party 2, and x_{i3} to party 3).

- Third, each party computes the addition of the shares locally:

  Party 1 computes: $x_{11} + x_{21} + x_{31} = 34$

  Party 2 computes: $x_{12} + x_{22} + x_{32} = -7$

  Party 3 computes: $x_{13} + x_{23} + x_{33} = -6$

- Finally, each party can reveal its result and add them jointly to get the actual output (i.e., $34 - 7 - 6 = 21$).

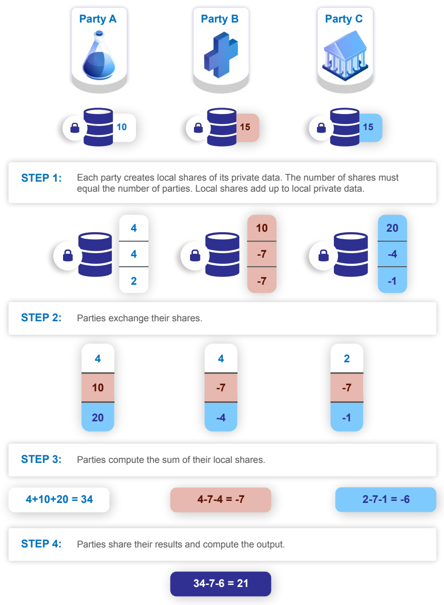

Party A
Party B
Party C
10
15
15
STEP 1: Each party creates local shares of its private data. The number of shares must equal the number of parties. Local shares add up to local private data.
4
4
2
10
-7
-7
20
-4
-1
STEP 2: Parties exchange their shares.
4
10
20
4
-7
-4
2
-7
-1
STEP 3: Parties compute the sum of their local shares.
4+10+20 = 34
4-7-4 = -7
2-7-1 = -6
STEP 4: Parties share their results and compute the output.
34-7-6 = 21

Figure 1: MPC Addition Example

The above example demonstrates that secure MPC allows different parties to jointly compute a target function without revealing any of their data. Note that all exchanged shares are random number that completely preserve the privacy of the original data. Similarly, to compute multiplications and AND gates, we use a technique invented by Beaver[21]. We also notice from the example that evaluating the addition function does not require additional communications between the parties. In contrast, secure MPC protocols require a single round of communication *for each* multiplication or AND processes. Thus, the secure MPC protocol introduces communication overhead due to the large number of communications required to evaluate some complicated functions.

3 System Design and Implementation

We detail in this section the overall system design and implementation with a focus on the Blind Inference service, based on the following real-world scenario: *Party A* is a healthcare provider that built a deep learning model for predicting the patients sex using their electrocardiogram (ECG) data. We refer to Party A as the *model owner* or the *service provider* (aka., server). Party A wishes to test the model on out-of-sample data, but unfortunately they do not have access to such test data. *Party B*, another healthcare provider, owns a large number of ECG samples with their gourd-truth labels (patient's sex). Party B is referred to as the data owner (aka., client). Parties A and B are open for collaboration but are unwilling and *cannot* share neither the model (due to IP concerns) nor the test data (due to privacy and legal concerns). Therefore, a practical solution here is to use Blind Inference, which allows both parties to securely complete their collaboration without revealing neither the model nor the data. First, we assume that both parties install our system as a docker container on their cloud or local machines (we refer to this machine as *Access Point*). Then, using our Router's and web application, the involved parties can use automated APIs to establish agreements, run and manage the prediction tasks, and audit the process. Our Router only facilitates the management and initiation of the tasks and does "see" any of the parties' data.

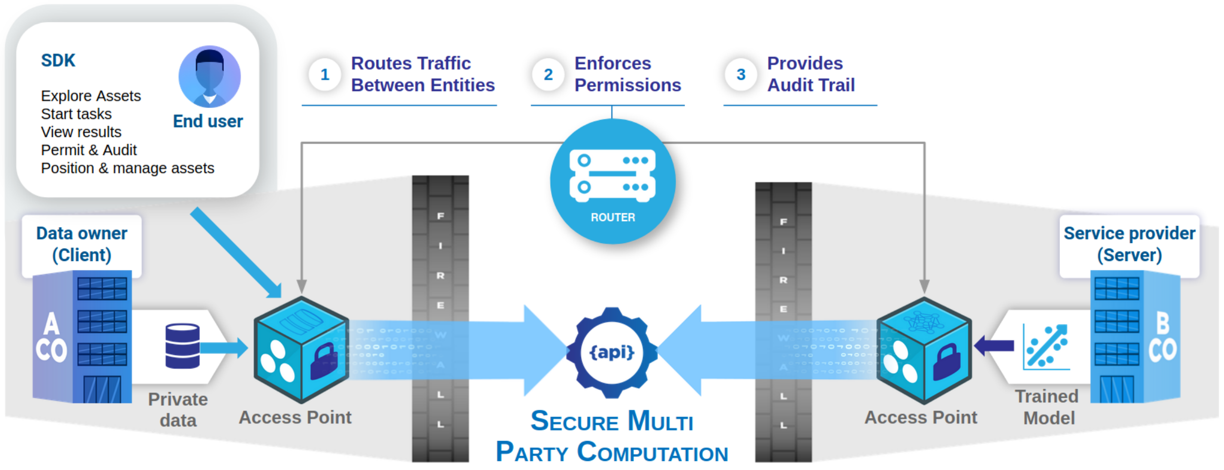


Figure 2: An Overview of TripleBlind's Secure MPC inference System.

3.1 Access Point

An Access Point (AP) is a docker container running our software system within a dedicated machine inside the organization's (user) infrastructure (depicted in Blue boxes in Figure 2). It is used to host the organization's assets (i.e., data and algorithms) and connect the organization to our ecosystem.

AP's main feature is its ability to utilize the user's assets on-site and run our services on them without sending the user's data outside its infrastructure. APs of different organizations interact with each other directly using secure channel communications, which are managed by the Router, but never passes through the Router. APs use an *SQLite* database to store and manage their status and the running jobs. *Flask*, a Python web framework, is used to facilitate and manage the communications between the Access Points with the Router.

3.2 Router

The Router is the primary management unit of our ecosystem, deployed as a cloud service responsible for managing and coordinating the users, jobs (i.e., secure MPC inference), communications, permissions, and digital rights. It exposes a set of public APIs for coordinating the joint operations and a web user-interface (web-UI) for other management tasks. The web UI also allows users to browse and explore information about the assets they wish to index for monetary, collaboration, or research purposes. For example, some organizations can list information about their trained models and allow other organizations to benefit from them securely. For the authentication and authorization of the communications between Access Point-to-Router, SDK-to-Access Point, and Access Point-to-Access Point we use *JSON Web Token (JWT)*. Communications between the involved parties are carried out using the encrypted communication protocol, *HTTPS*. *Secure Socket Layer (SSL)* is used to verify the authenticity of all parties in the ecosystem and to encrypt communications between them.

The Router uses the *PostgreSQL* database to store, organize, and manage the indexed assets, permissions, agreements, jobs, and organizations' accounts. Indexed assets are managed by the owner organization either via the Routers web-UI or the SDK. The list of assets is stored in the Router's database, while the actual raw data of the asset remains at the owner's Access Point. The Router's web-UI is built using *React*, an open-source *JavaScript* library. The back-end of the Router is build using Python's web framework *Django*. To enable the Router APIs, we use the *Django REST* framework.

3.3 Software Development KIT (SDK)

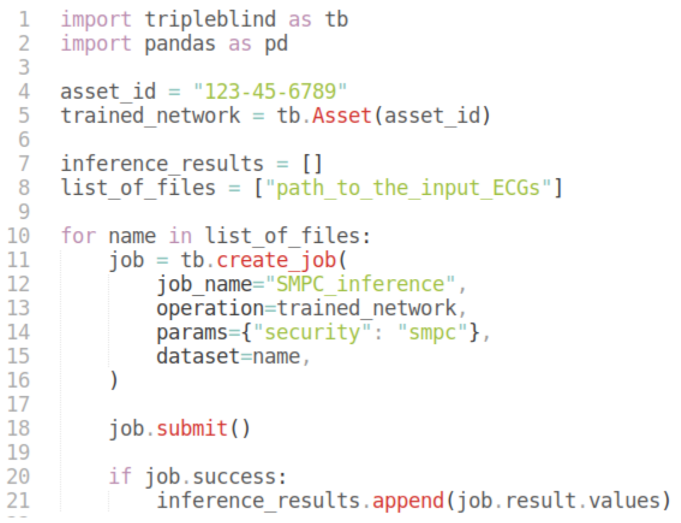
The SDK provides complete scripting control of our services, e.g., secure MPC inference, for the end-user. It is installed on the end user's device (e.g., a data scientist's workstation) to manage the organization's assets or operate on other organizations' assets for training, inferences, or analysis. The SDK includes a Python library and command-line utilities to interface with the rest of the ecosystem. We provide an end-to-end example of providing and consuming a deep learning model for secure inference using our toolset in the following subsection.

3.4 Example of Secure MPC Usage with Blind Inference

Figure 3 is the exact code run by the data owner to execute a secure MPC inference via a model owned by another party. Specifically, as shown in the figure, the data owner can query the private model using a straightforward function call and does not require the end-user to be familiar with the underlying cryptographic techniques. We assume that the model owner is also a user of Blind Inference and used the platform to index their model under the ID `123-45-6789`. The data owner first gets the model's ID, represented in Line 4, either from the model owner directly or using the Router's web application–which lists all assets made discoverable by their owners. Line 8 represents a list of the files that the user wants to query as an input to the model. It is important to specify the needed protocol (i.e., "smpc" as illustrated in Line 14). The function `job.submit()` will start the inference task using the specified input file and the model on hand using the secure MPC protocol. Finally, the user can collect the output results in any format (e.g., csv files, text files). In the example, we use a list to append the results of each inference, as shown in Line 21.

4 Evaluation

We evaluate in this section the efficiency and accuracy of our MPC toolset. Our overall experiments are divided into three experiments. **Experiment A:** we run a set of experiments that compares our work to the current state-of-the-art work in this domain following the conventional evaluation methods of MPC. The goal here is to evaluate our tool against several MPC protocols (algorithms). **Experiment B:** we evaluate the precision of the results produced by our system on a real-world medical application. This experiments shows that our approach outputs results that are identical to plain-text inference up to four decimal points. **Experiment C:** we evaluate the overall tool efficiency compared to two other tools, CrypTen and PySyft.



```
1   import tripleblind as tb
2   import pandas as pd
3
4   asset_id = "123-45-6789"
5   trained_network = tb.Asset(asset_id)
6
7   inference_results = []
8   list_of_files = ["path_to_the_input_ECGs"]
9
10  for name in list_of_files:
11      job = tb.create_job(
12          job_name="SMPC_inference",
13          operation=trained_network,
14          params={"security": "smpc"},
15          dataset=name,
16      )
17
18      job.submit()
19
20      if job.success:
21          inference_results.append(job.result.values)
```

Figure 3: A code snippet illustrating the usage of Blind Inference API

4.1 Evaluation Methodology and Settings

Experiment A: we compare and contrast the efficiency and accuracy of our MPC protocol to the prior work in this domain. The efficiency is measured by the overall execution time (in seconds) that it takes to compute a single prediction. The accuracy is measured based on the number of matching outputs between the MPC prediction and the plain-text prediction. We compare our approach to MiniONN[22], Gazelle[17], Chameleon[16], and SecureNN[15]. Following the evaluation methods of prior work in this domain, we first use the well-known benchmark dataset MNIST[12] to train three types of neural networks, each trained for 15 epochs using the `Adam` optimizer with the default learning rate of 0.001 and a batch size 64. We used all $50,000$ images for the training and kept the $10,000$ test images for evaluating the prediction results and accuracy (see below). These experiments were carried out with the following models:

- Network-A: three fully-connect layers with ReLU activation functions. It achieved 97.60% accuracy.
- Network-B: a three-layer network made of one convolution (Conv-2D) layer with 1 input channel, 16 output channels, and a 2×2 Maxpool followed by a ReLU and 2 fully-connected layers. This model achieved 98.55% accuracy on the test set.
- Network-C: a 4-layer network made of 1 Conv-2D with 1 input channel, 20 output channels and a 5×5 filter. The activation functions following this layer are ReLU, followed by a 2×2 Maxpool. The second layer is a Conv-2D layer with 20 input channels, 50 output channels, and a 5×5 filter followed by a ReLU and a 2×2 Maxpool. The third layer is an 800×500 fully-connected layer. The next activation function is ReLU. The final (output) layer is a 500×10 linear layer. This model achieved 99.47% accuracy on the test set.

Experiment B: we illustrate the prediction precision (confidence score) of our MPC method up to four decimal points based on the results generated by a real use case that utilized our toolset. The goal of the use case was to run prediction tasks on a pretrained model to infer the subject's sex using their ECG. The model architecture is proprietary to the model developer, but we focus here on the prediction accuracy using this model when applied in a real world scenario. The authors of the model used the ECGRDVQ PhysioNet database[23,24], which contains multi-channel ECG recordings of 22 healthy subjects partaking in a randomized, double-blind, 5-period crossover clinical trial aimed at comparing the effects of four known QT prolonging drugs versus placebo samples.

The data on these subjects also includes demographic information such as age, height, weight, and race for each record. A sample record of 12 channels is visualized in Figure 4. During each period, continuous ECGs were recorded at 500 Hz with an amplitude resolution of 2.5 V. From the continuous recording, triplicate 10-second ECGs were extracted at 16 predefined time-points. At each of the 16 time-points, three optimal 10-second 12-lead ECGs were extracted with stable heart rates and maximum signal quality. The resulting 5,232 ECGs were up-sampled from 500 to 1000 Hz.

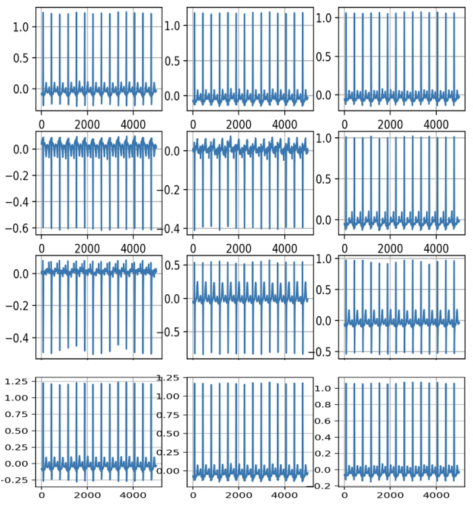


Figure 4: A sample ECG record.

Experiment C: we investigate the overall efficiency of our tool compared to the most popular tools in this domain, namely PySyft and CrypTen. For this experiment, we use three models on two datasets: Network-A and Network-B from Experiment A, and Netwok-D on a Malaria Infection dataset[25]. Network-D is based on the LeNet-5 architecture and comes very similar to Network-C with with added number of output channels and larger dense layers, based on the network proposed in[22]. Network-D was trained for on the Malaria Infection dataset using $24,804$ images and achieved 94.85% accuracy on $2,756$ test images, where all images were resized to 32×32 to match the prior work results.

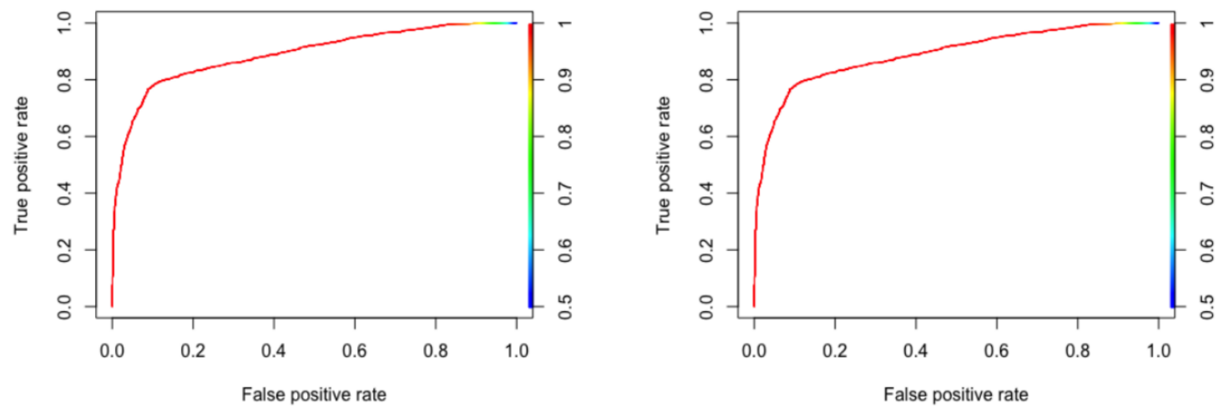
Table 1: Execution time comparison of different MPC methods for an inference task over a LAN. Time in *seconds*.

Approach $\rightarrow$	**Blind Inference (ours)**	SecureNN	Gazelle	MiniONN	Chameleon
Network A	**0.023**	0.043	0.09	1.04	-
Network B	**0.025**	0.076	0.29	1.28	2.7
Network C	**0.036**	0.13	1.16	9.32	2.24

4.2 Results

Experiment A. Table 1 lists the execution time results of our approach compared to five other methods, including SecureNN[15], Gazelle[17], MiniONN[22], and Chameleon[16]. The experiment results illustrate that our approach is the most efficient among the existing work and is applicable to realistic models that were not evaluated in these works (see Experiment B). It is also evident from the results that our protocol imposes less overhead with larger models. For example, SecureNN protocol introduces almost $2\times$ latency between Network-B and Network-C, which only differ in the size of their layers. In contrast, our protocol is more than $3.5\times$ faster than SececureNN–considered the state-of-the-art of MPC-based prediction.

Experiment B. We evaluated the accuracy of a pre-trained model using the standard (plain-text) and secure MPC inference approaches on the ECGRDVQ PhysioNet dataset. We compared the ROC curves, the area under ROC curves, and the confusion matrices across both models. The ROC curves that plot the false positive rate versus the true positive rate are highly similar across both approaches, as shown in Figure 5. The areas under the ROC curves were highly similar up to 4 decimal places across these models, with 0.8997362 for the secure MPC inference compared to 0.8997434 for the standard model inference. Additionally, Figure 6 shows that the point estimates of the prediction probabilities obtained from secure MPC versus standard models show an exact match in predictions between both models; i.e., 100% match in accuracy results. The confusion matrices were also the same over the secure and non-secure models with $4,713$ correct classifications and 519 wrong classifications. A careful examination of the floating-point values of these estimates to determine the precision in the match shows that these probabilities match up to 4 decimal places.



(a) standard inference (b) secure MPC inference

Figure 5: ROC's of the standard and secure MPC inference results. Both curves look identical and the areas under ROC curves are quite similar with 0.8997434 (standard) versus 0.8997362 (Blind Inference).

Experiment C We can observe from Table 2 that our work, based on the design and implementation details introduced in Section 2, presents the most efficient MPC toolset. We attribute the discrepancy in run times to the use of different implementation decisions, such as communication backends, serialization and deserialization, and the overall tool ecosystem. In contrast, PySyft represent the easiest tool to download and start a job compared to both CrypTen and our toolset but at the cost of significant execution latency.

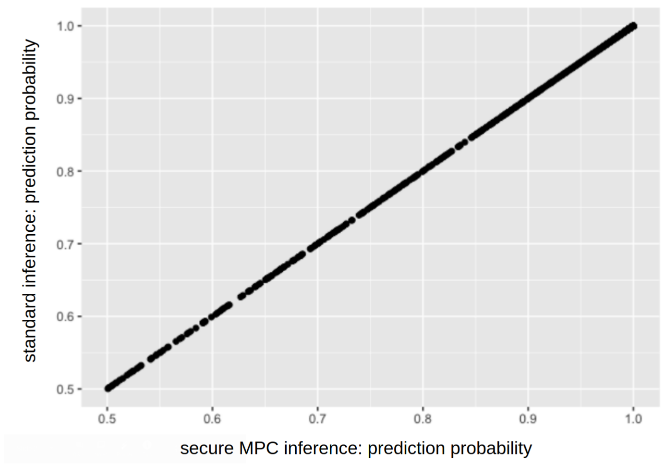


Figure 6: A plot representing the point estimates of the prediction probabilities obtained from Blind Inference versus standard inference. The plot shows an exact match in predictions between both inference results.

Table 2: Execution time comparison of different MPC inference tools. Time in *milliseconds*.

Tool	PyTorch	CrypTen	PySyft	**Blind Inference (ours)**
Network-A	0.2	43	500	**22**
Netwokr-B	0.3	35	660	**25**
Network-D	1.1	250	8,200	**170**

5 Prior Work

MPC tool support. Some tools have recently evolved to address the lack of tool support for MPC-based inference systems, such as CrypTen from Facebook and PySyft from OpenMined. However, CrypTen does not support the Windows operating system currently and is limited to models developed in PyTorch. Moreover, CrypTen is not a complete system for MPC inference, it only provides the necessary library on top of which other systems could be built. In contrast, our tool supports all operating systems and models developed in TensorFlow, Keras, and *PyTorch*, and it is a complete toolset. Additionally, while PySyft represent a very promising open-source solution, it is still considered a research tool and lacks critical features to become adopted in actual application, such as digital agreements and auditing. It also supports models build in *PyTorch* only and, as we demonstrated in our experiments, imposes significant execution overhead that makes it undesirable for real usage.

Homomorphic encryption. Homomorphic encryption, and specifically fully homomorphic encryption gentry2009fully, can be used to preserve the privacy of the data by encrypting it with a key while preserving the structure of the underlying operations (e.g., predicitom). The user data is encrypted on the client machine and then sent to the server where the actual computations take place. Examples include n-Graph boemer2019ngraph, which does not support rectified linear units (ReLU) in the neural network or other complicated activation functions; CryptoNets[26], which is one of the first works that utilized FHE for secure inference, and several others works that built on it[27,28]. While FHE can preserve the privacy of the data via encryption, it still faces several challenges in the context of deep learning models. For example, the supported operations on the encrypted data are limited to addition and multiplications, while other operations such as activation functions are often approximated using polynomials, which can result in significant accuracy degradation. Another critical limitation that prevents FHE from being widely adopted is its expensive computations and latency. For example, executing the same prediction with Network-D using FHE will require over 548 seconds[29] compared to 170 milliseconds in our approach.

Trusted execution environments (confidential computing). Hardware based methods for privacy-preserving inference have gained much attention recently[30]. Secure enclaves[31] enable confidential computing, a process that ensures that different programs running on the same machine or cloud server cannot access one another's memory, keeping data in use private. Confidential computing relies on the usage of secure enclaves such as Intel Software Guard Extensions (SGX) and provides cryptographic proof to compute in a secure container within an untrusted machine through memory isolation and encryption.

While secure enclaves can aid data privacy issues related to keeping data private from others with access to the same physical hardware on a public cloud, they still face several challenges, including that secure enclaves do not allow operations on European data to take place from the US and the currently available secure enclaves maintain low secure Processor Reserve Memory (PRM) that is not sufficient for even medium-sized models. For example, some of the Intel SGX maintains 128 MB of secure PRM, of which 90 MB is the Enclave Page Cache (EPC); simultaneously, a `ResNet-50` model has a size of 102 MB. Overall, most existing confidential computing solutions could offer better efficiency than secure MPC protocols; however, this efficiency comes at the price of a weaker threat model that requires trusting the hardware vendor and providing powerful defenses against secure enclave attacks[32,33].

Conclusions

We presented in this paper a the details of the design and implementation of a novel complete system that support automated MPC-based inference for neural networks. We specifically focused on facilitating and accelerating the secure inference service for non-cryptography experts, such as ML practitioners from physicians. We also demonstrated via our thorough experiments the efficiency (more than $3.5\times$ faster) and accuracy (100% matching predictions up to four decimal points confidence) for our toolset and its underlying algorithm compared to the state-of-the-art methods and tools. We hope that our details system design and implementation can spur the adoption of MPC-based method for privacy-preserving ML application.

Our main learning objectives include the following:

- *Learn* and the current challenges in inference-as-a-service tools and their limitations
- *Understand* the general concept of secure MPC and its benefits for prediction systems, specifically in the medical domain
- *Identify* proper methods for privacy-preserving applications, among Secure MPC, FHE, and trusted execution environments
- Apply an automated toolset for secure MPC prediction

References

1. Rajpurkar P, Chen E, Banerjee O, Topol EJ. AI in health and medicine. Nature Medicine. 2022:1-8.
2. Lopez-Jimenez F, Attia Z, Arruda-Olson AM, Carter R, Chareonthaitawee P, Jouni H, et al. Artificial intelligence in cardiology: present and future. In: Mayo Clinic Proceedings. vol. 95. Elsevier; 2020. p. 1015-39.
3. Castiglioni I, Rundo L, Codari M, Di Leo G, Salvatore C, Interlenghi M, et al. AI applications to medical images: From machine learning to deep learning. Physica Medica. 2021;83:9-24.
4. Horvitz E, Mulligan D. Data, privacy, and the greater good. Science. 2015;349(6245):253-5.
5. Al-Rubaie M, Chang JM. Privacy-preserving machine learning: Threats and solutions. IEEE Security & Privacy. 2019;17(2):49-58.
6. van Veen EB. Observational health research in Europe: understanding the General Data Protection Regulation and underlying debate. European Journal of Cancer. 2018;104:70-80.
7. Stallings W. Handling of Personal Information and Deidentified, Aggregated, and Pseudonymized Information Under the California Consumer Privacy Act. IEEE Security & Privacy. 2020;18(1):61-4.
8. Yao ACC. How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (sfcs 1986). IEEE; 1986. p. 162-7.
9. Gentry C, Boneh D. A fully homomorphic encryption scheme. vol. 20. Stanford University Stanford; 2009.
10. Naor M, Pinkas B. Oblivious transfer and polynomial evaluation. In: Proceedings of the thirty-first annual ACM symposium on Theory of computing; 1999. p. 245-54.
11. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998;86(11):2278-324.
12. LeCun Y, Cortes C. The MNIST dataset.;. (Last accessed 15-July-2021). Available from: `http://yann.lecun.com/exdb/mnist`.
13. Boemer F, Lao Y, Cammarota R, Wierzynski C. nGraph-HE: a graph compiler for deep learning on homomorphically encrypted data. In: Proceedings of the 16th ACM International Conference on Computing Frontiers; 2019. p. 3-13.

14. Knott B, Venkataraman S, Hannun AY, Sengupta S, Ibrahim M, van der Maaten LJP. CrypTen: Secure Multi-Party Computation Meets Machine Learning. In: Proceedings of the NeurIPS Workshop on Privacy-Preserving Machine Learning; 2020. .

15. Wagh S, Gupta D, Chandran N. SecureNN: 3-Party Secure Computation for Neural Network Training. Proc Priv Enhancing Technol. 2019;2019(3):26-49.

16. Riazi MS, Weinert C, Tkachenko O, Songhori EM, Schneider T, Koushanfar F. Chameleon: A hybrid secure computation framework for machine learning applications. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security; 2018. p. 707-21.

17. Juvekar C, Vaikuntanathan V, Chandrakasan A. {GAZELLE}: A low latency framework for secure neural network inference. In: 27th {USENIX} Security Symposium ({USENIX} Security 18); 2018. p. 1651-69.

18. Shamir A. How to share a secret. Communications of the ACM. 1979;22(11):612-3.

19. Ben-Or M, Goldwasser S, Wigderson A. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali; 2019. p. 351-71.

20. Goldreich O, Micali S, Wigderson A. How to play any mental game, or a completeness theorem for protocols with honest majority. In: Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali; 2019. p. 307-28.

21. Beaver D. Efficient multiparty protocols using circuit randomization. In: Annual International Cryptology Conference. Springer; 1991. p. 420-32.

22. Liu J, Juuti M, Lu Y, Asokan N. Oblivious neural network predictions via minionn transformations. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security; 2017. p. 619-31.

23. Goldberger AL, Amaral LA, Glass L, Hausdorff JM, Ivanov PC, Mark RG, et al. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. circulation. 2000;101(23):e215-20.

24. Johannesen L, Vicente J, Mason J, Sanabria C, Waite-Labott K, Hong M, et al. Differentiating drug-induced multichannel block on the electrocardiogram: randomized study of dofetilide, quinidine, ranolazine, and verapamil. Clinical Pharmacology & Therapeutics. 2014;96(5):549-58.

25. Rajaraman S, Antani SK, Poostchi M, Silamut K, Hossain MA, Maude RJ, et al. Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. PeerJ. 2018;6:e4568.

26. Gilad-Bachrach R, Dowlin N, Laine K, Lauter K, Naehrig M, Wernsing J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In: International conference on machine learning. PMLR; 2016. p. 201-10.

27. Brutzkus A, Gilad-Bachrach R, Elisha O. Low latency privacy preserving inference. In: International Conference on Machine Learning. PMLR; 2019. p. 812-21.

28. Sanyal A, Kusner M, Gascon A, Kanade V. TAPAS: Tricks to accelerate (encrypted) prediction as a service. In: International Conference on Machine Learning. PMLR; 2018. p. 4490-9.

29. Haralampieva V, Rueckert D, Passerat-Palmbach J. A systematic comparison of encrypted machine learning solutions for image classification. In: Proceedings of the 2020 workshop on privacy-preserving machine learning in practice; 2020. p. 55-9.

30. Karl R, Takeshita J, Jung T. Using Intel SGX to improve private neural network training and inference. In: Proceedings of the 7th Symposium on Hot Topics in the Science of Security; 2020. p. 1-2.

31. Costan V, Lebedev I, Devadas S, et al. Secure processors part I: background, taxonomy for secure enclaves and Intel SGX architecture. Now Foundations and Trends; 2017.

32. Murdock K, Oswald D, Garcia FD, Van Bulck J, Gruss D, Piessens F. Plundervolt: Software-based fault injection attacks against Intel SGX. In: 2020 IEEE Symposium on Security and Privacy (SP). IEEE; 2020. p. 1466-82.

33. Lipp M, Kogler A, Oswald D, Schwarz M, Easdon C, Canella C, et al. PLATYPUS: Software-based power side-channel attacks on x86. In: IEEE Symposium on Security and Privacy (SP); 2021. .