Proceedings of Machine Learning Research 176:343–348, 2022 NeurIPS 2021 Competition and Demonstration Track

TripleBlind: A Privacy-Preserving Framework for Decentralized Data and Algorithms

Gharib Gharibi
Babak Poorebrahim Gilkalaye
Ravi Patel
Andrew Rademacher
David Wagner
Jack Fay
Gary Moore
Steve Penrod
Greg Storm
Riddhiman Das
TripleBlind, Inc. RESEARCH@TRIPLEBLIND.AI

Editor: Douwe Kiela, Marco Ciccone, Barbara Caputo

Abstract

Centralized sharing of sensitive data for training and inference is challenging and undesired due to privacy, competition, and legal concerns. While distributed learning and secure inference have demonstrated significant privacy gains, these methods still largely ignore the design and implementation of practical, privacy-preserving tool support. To address these challenges, we present TripleBlind, an automated privacy-preserving framework for creating and consuming data-driven applications from decentralized data and algorithms. TripleBlind provides a set of automated, high-level APIs that enable (1) extracting knowledge from remote data without moving it outside the owner's infrastructure, (2) training AI models from decentralized data, and (3) consuming trained models for secure inference-as-a-service; all without compromising the privacy of either the model/query or the data. In this short paper, we shed light on the underlying training and inference methods, the design and implementation of our framework, and showcase the actual code necessary to run a secure, remote inference using our secure multi-party computation API. A video demo highlighting the main features of our framework is located at www.tripleblind.ai/neurips2021

Keywords: Machine learning, deep learning, privacy, distributed learning, secure multi-party computation, inference-as-a-service

1. Motivation and Prior Work

Artificial intelligence (AI), especially deep learning, has led to significant advances in an ever-growing number of domains (Lopez-Jimenez et al., 2020; Lotter et al., 2021). The common denominator for this success heavily relies on the availability of large amounts of diverse data–which is critical for training high-performance, fair, and generalizable models. In contrast, training on single-origin, limited data can lead to poor models [1]. For example,

1. Note that some ML algorithms might not require training data to make correct predictions on a new task such as *zero-shot learning (Xian et al., 2018)*, but such algorithms were previously trained on similar tasks using large amounts of diverse data.

Driggs et al. (2021) investigated more than 400 models for detecting COVID-19 and determined that each one of them was fatally flawed when tested on out-of-sample data–mainly because they were trained on small, single-origin samples with limited diversity. The study highlights that *it is imperative to ensure easy and rapid access to diverse data without compromising its privacy for critical and high-quality data-driven research.* Thus, there exist pressing demands for (1) sharing multi-institutional and multi-national data to train high-performance AI models and (2) utilizing such models for inference services without harming the data privacy or the model's intellectual property. Unfortunately, it is still profoundly challenging to share and utilize such sensitive data and models due to privacy (Shokri et al., 2017; Nasr et al., 2019; Rigaki and Garcia, 2020; Zhang et al., 2020), ethical (Krupinski, 2020; Morley et al., 2020), and legal concerns CCPA (2018); GDPR (2016).

First, to address private data sharing, *distributed learning methods* that enable training a shared model from decentralized data have emerged, such as Federated Learning (`FL`) (McMahan et al., 2017), Split Learning (`SL`) (Vepakomma et al., 2018), and SplitFed Learning (`SplitFed`) (Thapa et al., 2020). Second, to address inference-as-a-service privacy challenges, several studies suggested the use of cryptographic methods, specifically *secure multi-party computation (`MPC`)* (Yao, 1986), which has illustrated promising results (Juvekar et al., 2018; Riazi et al., 2018; Wagh et al., 2019). Nevertheless, the majority of the existing work in this domain largely ignore the design and implementation of proper tool support for these methods. For example, most of the distributed learning tools are still experimental, including *FedML* (He et al., 2020) and *Tensorflow Federated* (Google, 2019). Similarly, the majority of secure `MPC` inference tools are either experimental, such as *CryptFlow* (Kumar et al., 2020), or require deep understanding of the underlying cryptographic protocols. While *CrypTen* (Knott et al., 2021) and *PySyft* (Ziller et al., 2021) address the latter issue, they are both specific to *PyTorch* and do not provide a complete privacy-preserving software system. *PySyft*'s implementation also adds significant latency when used over the public Internet (see our preliminary results).

2. TripleBlind: System Overview

Figure 1 illustrates an overview of our system's architecture, made of three major components: the Router, Access Point (AP), and a Software Development Kit (SDK). We explain these components and a typical workflow of the system using the following real-world use case. We assume all parties have our software framework installed on their machines.

2.1. Use case

Alice is a physician and a deep learning practitioner at `Hospital A`, who developed a high-performing model to predict a specific disease. She is interested in validating her model on out-of-sample data which she does not own, but is available at `Hospital B`. However, `Hospital B` cannot simply share their patients' data with Alice due to privacy and legal concerns. Meanwhile, Alice is concerned about the intellectual property of her model, so she is unwilling to share it with `Hospital B` to validate it locally on their data. Using *TripleBlind's* secure `MPC` inference protocol, Alice can validate her model on `Hospital B`'s data without exposing neither the model nor the test data. Following, we illustrate the workflow of this task and then explain the system components in the next subsection.

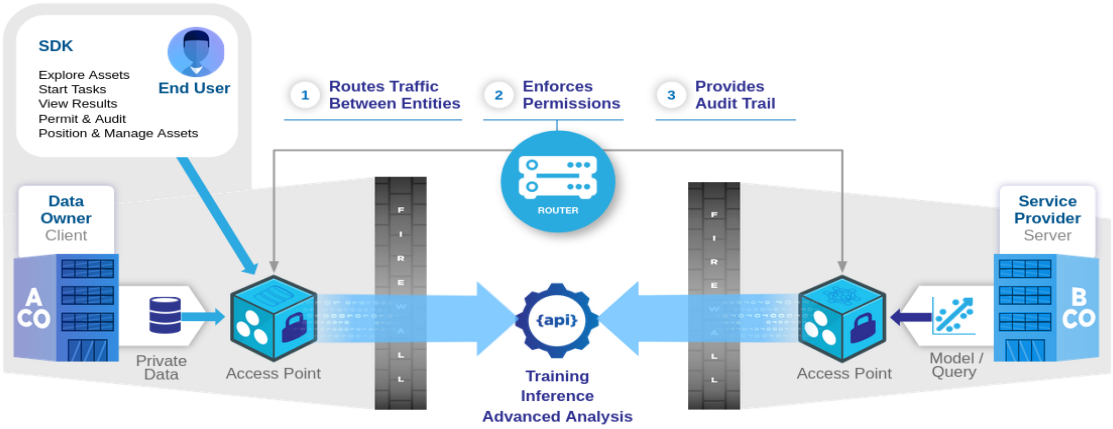

Figure 1: TripleBlind System Overview

First, both parties position their assets (i.e., model and data) on our software platform, which is installed as a docker container on their local or cloud machines. Note that both assets are still physically stored on the users' machines–and never leave them. We refer to these machines as Access Points, visualized as Blue boxes in Figure 1. The asset owner can also *list* the asset in the Router to allow other organizations to discover it for collaboration, either using our SDK or via the Router's web interface. Figure 2 illustrates the actual code snippet necessary to position an asset (e.g., Alice's trained model) using the SDK.

Second, Alice uses our automated API to run the secure inference on `Hospital B`'s data. Figure 3 illustrates the usage of the API. Line 2 uses the public ID of `Hospital B` data, which Alice discovered using the Router's web interface. Third, `Hospital B` will receive an access request to use their data. Upon reviewing the request and then approving it, our protocol will take place between both parties directly while providing live monitoring and audit. Finally, Alice will receive the inference results in a zipped file (Line 14), which she can use to obtain different validation metrics.

2.2. System Components

The Access Point (AP) is a Docker container running our software system within a dedicated machine inside the organization's infrastructure. The AP hosts the organization's assets, connects them to our ecosystem, and utilizes the assets for joint operations (e.g., distributed learning, inference) without sending any raw data outside the user's infrastructure. The APs of different organizations interact with each other directly using secure channel communications, which are managed by our Router but never pass through it.

The Router is responsible for managing and coordinating the users, operations, communications, permissions, and digital rights. It uses *JSON Web Token (JWT)* for the authentication and authorization of communications between the APs and the Router. Communications between the involved parties are carried out using the encrypted communication protocol *HTTPS* while the *Secure Socket Layer (SSL)* is used to verify the authenticity of all parties and to encrypt the communications between them. The Router stores a list of all organizations' assets, but the actual assets remain at the owner's AP.

```python
import tripleblind as tb

model = tb.asset.neural_network.create(
    path= "Alice_trained_model.pth",
    name="model_name",
    desc="Model to diagnose EKG.." )

model.add_agreement(
    with_org="Hospital_B",
    operation=tb.Operation.EXECUTE)
```

Figure 2: Asset Positioning API

```python
test_data = tb.Asset("Hospital_B_data_ID")
job = tb.create_job(
    operation=model,
    dataset=[test_data],
    params={
        "security": "smpc",
        "data_type": "numpy",
        "final_layer_softmax": True})

if job.submit():
    job.wait_for_completion()
    if job.success:
        filename = job.result.download("results.zip")
```

Figure 3: Blind Inference API

The Software Development KIT (SDK) is installed on the end user's device (e.g., a data scientist's workstation) and provides complete scripting control of our framework. It is used to manage the organization's assets or run joint operations, such as distributed training, inferences, or analysis. The SDK supports Python, R, and provides command-line utilities to interface with the rest of the ecosystem.

3. TripleBlind Training and Inference Algorithms

Blind Learning (`BL`): A distributed learning approach that introduces several novelties over `FL`, `SL`, and `SplitFed`. `BL` splits the model's training process between the server and clients, thus reducing the computational burdens at the clients compared to `FL`. `BL` trains the clients' models in parallel leading to more efficient training compared to `SL`. It also utilizes a weighted average loss function at the server-side that requires a single update per each communication round, leading to better performing models than both `SL` and `SplitFed`. For example, training a shared `ResNet-34` model on the `CIFAR-10` dataset divided among 3 clients in a practical non-IID distribution requires 60% less computations in `BL` compared to `FL` to reach a baseline accuracy of 80%. Both `SL` and `SplitFed` fail to reach this accuracy in the allowed number of communication rounds in this experiment.

Blind Inference: A secure multi-party computation protocol for automated inference-as-a-service built using *secret sharing techniques*. It operates in a semi-honest setting, in which an adversary tries to learn information about the other party's data but never deviates from the protocol. Our inference also enforces an *agreement* phase before the computation begins, in which both parties decide on the input data, the results to disclose, and policies around the task. Blind Inference can compute an inference using `LeNet-5` trained on `MNIST` in 0.036 seconds compared to 0.13 and 1.16 seconds in *SecureNN* (Wagh et al., 2019) and *Gazelle* (Juvekar et al., 2018), respectively.

4. Conclusions

There is an evident lack of studies that detail the design and implementation of software systems for privacy-preserving methods and tools. We explained in this paper our design and implementation of a practical privacy-preserving framework that allows providing and consuming data-driven applications over the public Internet. We hope our implementation techniques will spur the adaption and implementation of secure and private AI systems.

References

CCPA. California consumer privacy act, 2018. URL https://oag.ca.gov/privacy/ccpa.

Derek Driggs, Ian Selby, Michael Roberts, Effrossyni Gkrania-Klotsas, James HF Rudd, Guang Yang, Judith Babar, Evis Sala, Carola-Bibiane Schönlieb, and AIX-COVNET collaboration. Machine learning for covid-19 diagnosis and prognostication: lessons for amplifying the signal while reducing the noise, 2021.

GDPR. Regulation eu 2016/679 of the european parliament and of the council of 27 april 2016. *Official Journal of the European Union*, 2016.

Google. TensorFlow Federated. https://www.tensorflow.org/federated/, 2019. Online; accessed 15 November 2021.

Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, et al. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020.

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

Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. Crypten: Secure multi-party computation meets machine learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Elizabeth A Krupinski. An ethics framework for clinical imaging data sharing and the greater good, 2020.

Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow: Secure tensorflow inference. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 336–353. IEEE, 2020.

Francisco Lopez-Jimenez, Zachi Attia, Adelaide M Arruda-Olson, Rickey Carter, Panithaya Chareonthaitawee, Hayan Jouni, Suraj Kapa, Amir Lerman, Christina Luong, Jose R Medina-Inojosa, et al. Artificial intelligence in cardiology: present and future. In *Mayo Clinic Proceedings*, volume 95, pages 1015–1039. Elsevier, 2020.

William Lotter, Abdul Rahman Diab, Bryan Haslam, Jiye G Kim, Giorgia Grisot, Eric Wu, Kevin Wu, Jorge Onieva Onieva, Yun Boyer, Jerrold L Boxerman, et al. Robust breast cancer detection in mammography and digital breast tomosynthesis using an annotation-efficient deep learning approach. *Nature Medicine*, 27(2):244–249, 2021.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

Jessica Morley, Caio CV Machado, Christopher Burr, Josh Cowls, Indra Joshi, Mariarosaria Taddeo, and Luciano Floridi. The ethics of ai in health care: a mapping review. *Social Science & Medicine*, 260:113172, 2020.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

M Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M Songhori, Thomas Schneider, and Farinaz Koushanfar. Chameleon: A hybrid secure computation framework for machine learning applications. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 707–721, 2018.

Maria Rigaki and Sebastian Garcia. A survey of privacy attacks in machine learning. *arXiv preprint arXiv:2007.07646*, 2020.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

Chandra Thapa, Mahawaga Arachchige Pathum Chamikara, Seyit Camtepe, and Lichao Sun. Splitfed: When federated learning meets split learning. *arXiv preprint arXiv:2004.12088*, 2020.

Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

Sameer Wagh, Divya Gupta, and Nishanth Chandran. Securenn: 3-party secure computation for neural network training. *Proc. Priv. Enhancing Technol.*, 2019(3):26–49, 2019.

Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018.

Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167. IEEE, 1986.

Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 253–261, 2020.

Alexander Ziller, Andrew Trask, Antonio Lopardo, Benjamin Szymkow, Bobby Wagner, Emma Bluemke, Jean-Mickael Nounahon, Jonathan Passerat-Palmbach, Kritika Prakash, Nick Rose, et al. Pysyft: A library for easy federated learning. In *Federated Learning Systems*, pages 111–139. Springer, 2021.