Cryptographic Auditing for Collaborative Learning

Hidde Lycklama, Nicolas Küchler, Alexander Viand
Emanuel Opel, Lukas Burkhalter, Anwar Hithnawi

ETH Zürich

Abstract

Collaborative machine learning paradigms based on secure multi-party computation have emerged as a compelling alternative for sensitive applications in the last few years. These paradigms promise to unlock the potential of important data silos that are currently hard to access and compute across due to privacy concerns and regulatory policies (e.g., health and financial sectors). Although collaborative machine learning provides many privacy benefits, it makes sacrifices in terms of robustness. It opens the learning process to the possibility of an active malicious participant who can covertly influence the model's behavior. As these systems are being deployed for a range of sensitive applications, their robustness is increasingly important. To date, no compelling solution exists that fully addresses the robustness of secure collaborative learning paradigms. As the robustness of these learning paradigms remains an open challenge, it is necessary to augment these systems with measures that strengthen their reliability at deployment time. This paper describes our efforts in developing privacy-preserving auditing mechanisms for secure collaborative learning. We focus on audits that allow tracing the source of integrity issues back to the responsible party, providing a technical path toward accountability in these systems.

1 Introduction

Machine learning algorithms continue to achieve remarkable success in a wide range of applications. This success, however, has been limited to domains and sectors where suitable training data is accessible. However, many of today's societally important questions require sensitive data that is hard to access and share due to privacy concerns, conflicting business interests, and/or regulatory restrictions [13, 14]. Collaborative Learning [38, 12, 28, 34, 39, 24] (CL) has recently emerged as a compelling solution for this setting. CL enables organizations to train a machine learning model on joint datasets without revealing their sensitive data. To achieve this, these systems employ cryptographic techniques such as secure Multi-Party Computation (MPC) that ensure the confidentiality of participants' input data and reveal only the final trained model. Although they provide many privacy benefits, these systems exacerbate existing ML robustness issues: they open the training process to potentially malicious parties which can covertly introduce maliciously crafted data in order to influence the model behavior. While data poisoning is not a new challenge, the privacy-protections of this setting enable a class of covert attacks and exclude many of the existing countermeasures.

Currently, we lack comprehensive solutions to these integrity issues in this setting; proposed defenses are repeatedly being broken [3, 10, 11, 26, 8] and more powerful attacks continue to surface [31, 37]. There is a growing realization in the community that we need to go beyond attack prevention and think of broader measures to complement existing defenses to minimize the consequences of these attacks in practical deployments [38, 23, 32]. A key complementary technique to existing defenses is *auditing* for accountability, allowing the model provider to identify the source of unwanted behavior. For systems trained on sensitive data, these measures need to be designed with caution to preserve

ML Safety Workshop at the 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

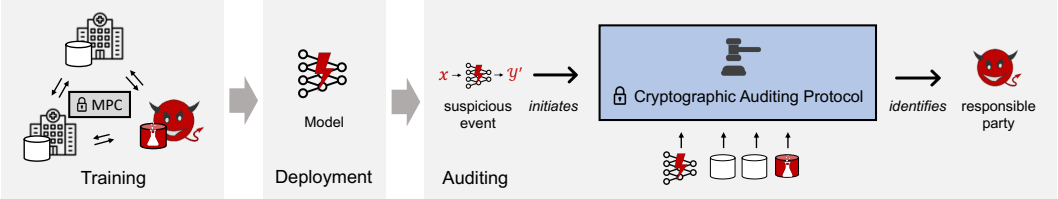

Figure 1: Privacy-preserving auditing scenario, identifying the source of a suspicious event.

the privacy of honest participants. In addition, they also need to avoid implicating honest clients, especially if the unwanted behavior is not actually caused by an attack. In this work, we present a novel privacy-preserving auditing technique to enable accountability in secure collaborative learning systems.

Contributions: In this paper, we augment secure CL with privacy-preserving audits for accountability. We first define auditing in secure collaborative learning settings and introduce an extensible framework for auditing; we describe the workflow of the framework and describe key cryptographic and algorithmic building blocks. Next, we propose an efficient auditing function that traces back abnormal behavior (i.e., misclassifications) to the responsible party. Our audit function uses approximate unlearning to identify each party's influence on a certain inference. Our evaluation shows that we can achieve high recall while guaranteeing a precision of at least 0.99 across a wide range of attacks and datasets. We show that we can efficiently extend secure collaborative learning with effective privacy-preserving audits without harming the privacy of benign participants.

2 Cryptographic Auditing

We begin by defining the scenario for privacy-preserving auditing, discuss key requirements, and describe our threat model. Afterwards, we present our cryptographic auditing protocol and concrete instantiations of the auditing function for identifying the source of data poisoning attacks.

2.1 Overview

Scenario. The standard CL setting features N parties training an ML model (classifier) $\mathcal{M}$ on their joint dataset $D = \cup_{i=1}^{N} D_i$, using MPC to ensure the confidentiality of the input data. Cryptographic auditing occurs after the resulting model is deployed when suspicious behavior is detected, e.g., a downstream incident tracing back to a suspicious model output $\tilde{y}$ for an input $\tilde{x}$ (a *misclassification event*). See Figure 1 for an overview of the scenario. The parties then engage in a cryptographic auditing protocol $\mathcal{P}$ initiated by the model operator. The privacy-preserving protocol takes as input the model $\mathcal{M}$, the misclassification event ($\tilde{x}$,$\tilde{y}$) and the parties' original training data. The parties execute an auditing function $\mathcal{F}$ that outputs the parties $\{ i \mid \text{party } i \text{ is responsible} \}$ that provided poisoned data leading to the misclassification event ($\tilde{x}$,$\tilde{y}$), or $\perp$ if no malicious activity is identified.

Key Requirements. Auditing on sensitive data needs to fulfill several key requirements: *(i) Privacy-Preserving.* Auditing must not weaken the strong privacy guarantees of collaborative learning. Therefore, auditing must not reveal the parties' sensitive input data and must especially not impact the privacy of honest parties. Parties should learn nothing from executing the protocol $\mathcal{P}$ beyond the output of the auditing function $\mathcal{F}$. In order to prevent harm to honest parties, the auditing function must have a low false-positive rate, since the consequences of being wrongly flagged as malicious might have implications for the privacy of a party. *(ii) Efficiency.* To be of practical use, the audit function must run efficiently on large-scale datasets. Specifically, running the auditing protocol cannot assume computation and bandwidth resources greater than needed for the original secure collaborative training. Since secure collaborative training is expensive due to the overhead of the employed secure computation techniques, the auditing should ideally be significantly cheaper.

Threat Model. We consider an adversary that compromises up to half of the parties by poisoning their dataset. Note that the guarantees of maliciously secure collaborative learning prevent attacks beyond poisoning and guarantee the confidentially of the input data. The adversary can modify any portion of the datasets $\tilde{D}_i \subseteq D_i$ of these parties, up to their full dataset, with the goal of making

the model misbehave at inference time. Note that we assume the adversary is aware of the auditing system in place and can actively attempt to fool the auditing algorithm. We employ maliciously secure MPC with guaranteed output delivery in an honest-majority setting [18, 15] for our auditing protocol, inheriting the assumptions and guarantees of these protocols. We rely on guaranteed output delivery to prevent the adversary from aborting the auditing protocol without being detected.

2.2 Cryptographic Auditing Protocol

The model operator triggers the cryptographic auditing protocol after detecting a suspicious misclassification event $\tilde{y} \leftarrow \mathcal{M}(\tilde{x})$. All the parties involved in the original training then engage in a privacy-preserving interactive MPC protocol to compute the auditing function $\mathcal{F}$. We require the party to commit to their data prior to training using cryptographic commitments [7, 29] and check the private input data provided to the auditing protocol against the commitments provided in the collaborative learning protocol. This ensures that the private input data provided when computing the audit function is the same as the data used during training and prevents malicious parties from altering or changing their data during the audit phase.

Towards this, we augment the secure training computation with a preprocessing step $\mathcal{V}$ that checks that the input data D_i for each party i corresponds to the commitments $\mathbf{c}_i$. The training computation proceeds only if these data consistency checks succeed. A straightforward implementation of this would require each party to broadcast their commitments to all other parties before training. However, if an MPC protocol that supports authenticated secret shares [16, 5] is used to instantiate both training and auditing, this preprocessing step can be omitted. During the auditing phase, we use the same preprocessing step $\mathcal{V}$ to verify that parties' commitments correspond to each party's original training data before evaluating the auditing function $\mathcal{F}$. Any party which provides inconsistent data is automatically flagged as malicious.

2.3 Auditing Function

We now present the design of an auditing function $\mathcal{F}$ to attribute responsibility for a misclassification event $(\tilde{x}, \tilde{y})$ to the responsible parties. The core of our technique is the computation of an *influence score* $\mathcal{S}_i$ for each party, where compromised parties will have a high influence score on poisoned data samples. However, a high influence score does not necessarily indicate maliciousness, as a misclassification event can arise even when no poisoning attack took place. In order to avoid harming benign parties, it is essential to distinguish these scenarios. We do so by considering the distribution of the influence scores: if a misclassification event was significantly influenced by a small subset of outlier parties, we assume a data poisoning attack. Conversely, if the influence is distributed evenly across all parties, we consider an attack less likely.

Thus, we want to identify which party's influence scores differ significantly from the overall population of scores. For each score, we compute the Median-Absolute-Deviation (MAD), a robust [1] measure of dispersion. We flag parties with a MAD score higher than a threshold τ as malicious. This threshold represents the maximum relative influence a party can have on a prediction before they are deemed suspicious and depends on the task, dataset and distribution shift between parties. Optimized values for τ could be derived from an analysis of a public (non-malicious) test set. However, we leave this to future work and choose conservative estimates for τ in our evaluation. In the following, we first present a baseline instantiation of an influence score using k-Nearest-Neighbors (kNN) and then our more sophisticated design using leave-out models.

k-Nearest-Neighbors. Using kNN, we can identify the k training samples closest to the prediction sample in the model's latent space. The influence score is then determined by the number of those k points contributed by a party. Formally, let Γ be the set of the k nearest neighbors to $\tilde{x}$, then $\mathcal{S}_i \leftarrow \text{count}\,(\Gamma \cap D_i)$. This is a simple and effective influence score if we assume that malicious training samples have small l_2-distances from $\tilde{x}$ in latent space. However, as we show in our evaluation, this assumption does not necessarily hold for attacks in practice.

Leave-out Models. The key idea for our design is that if a misclassification event was (at least partially) the result of poisoned data provided by a party; removing the data of that malicious party will result in the absence (or weakening) of the attack at inference time. We could train a leave-out

[1] Median-Absolute-Deviation is robust for distributions with up to 50% outliers, unlike its mean-based variant.

		Single Attacker				Multi Attacker			
		kNN		Unlearning		kNN		Unlearning	
		pre	rec	pre	rec	pre	rec	pre	rec
BN	CIFAR-10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
BN	Camelyon17	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99
EM	Ember	0.98	0.91	1.00	1.00	1.00	0.88	1.00	1.00
BP	CIFAR-10	0.79	0.97	1.00	1.00	1.00	0.58	1.00	0.53
BP	TinyImageNet	0.94	1.00	1.00	1.00	1.00	0.99	1.00	1.00
WIB	CIFAR-10	0.92	0.32	1.00	0.74	1.00	0.14	1.00	0.13

[1]BN: BadNets [20] EM: Ember Malware [31] BP: Bullseye Polytope [1] WIB: Witches' Brew [19]

Table 1: Performance of the kNN- and unlearning-based auditing functions for one and two attackers.

model $\mathcal{M}_i$ for each party, using the same dataset but *leaving out* party i's dataset. Given these, we can compute the influence score as the loss of each model on the misclassification event, i.e., $\mathcal{S}_i \leftarrow \ell(\mathcal{M}_i(\tilde{x}), \tilde{y})$. Note that, since the $\mathcal{M}_i$ do not depend on $\tilde{x}$ or $\tilde{y}$, we can use the same set of leave-out models to handle auditing requests for different events.

However, retraining these leave-out models naively is prohibitively expensive, as it requires training N additional models, each incurring the same costs as the original secure collaborative learning process. Therefore, our approach instead relies on techniques for *unlearning* a party's data points from the full model [9, 21, 6]. While *certified unlearning* approaches provide formal guarantees on their equivalence to leave-out models, these techniques currently do not scale well to scenarios like ours, where large amounts of data points must be removed [6, 36]. Instead, we use an efficient unlearning technique [32] that can scale to the demands of our setting. Specifically, $Unl(\mathcal{M}, D_i, D)$ applies first-order updates towards the objective $\min_\theta \left(\sum_{(x,y)\in D\setminus D_i} \ell\left(\mathcal{M}(x), y\right) + \sum_{(x,y)\in D_i} \ell\left(\mathcal{M}(x), \textit{EQUAL}\right) \right)$, where ℓ is the cross-entropy loss function and *EQUAL* is defined as the uniform probability vector, representing the output of the model when it is uncertain about its prediction [35, 27].

Our approach requires only a small number of unlearning steps, which can be orders of magnitude more efficient than full retraining. While the concrete number of steps required is task-dependent, in practice, our unlearning approach requires at most four epochs of training in settings where full training requires up to 200 epochs. Note that there are many other potential unlearning objectives, and a systematic investigation into optimal objectives for this task would be interesting future work. For brevity, we next focus our evaluation on the effectiveness of our auditing functions and omit a detailed performance evaluation of the MPC implementations.

3 Experiments

We evaluate the effectiveness and performance of cryptographic auditing on a wide range of attacks and datasets, where up to 40% of the client datasets are poisoned. We use the recommended configurations and settings from the data poisoning benchmark in [30] to evaluate auditing on the state-of-the-art Bullseye Polytope [1] (BP) and Witches' Brew [19] (WIB) clean label attacks as well as the dirty-label BadNets [20] (BN) attack. In addition, we also consider the clean-label Ember Malware attack [31] (EM). In contrast to dirty-label attacks that can poison samples arbitrarily, clean-label attacks rely on stealthy perturbations that are small enough to justify the original labels when observed by domain experts. In BP, the adversary inserts samples of the target class with small perturbations to form a polytope that traps the target inside the center of the samples' convex hull. In Witches' Brew, the adversary perturbs samples of the target class such that they mimic the gradient of the target labeled as the target class. We refer to Appendix §A for further details on the experiments.

We evaluate *(i)* the *precision* of identifying malicious clients from benign clients, and *(ii)* the *recall* of identifying malicious clients, i.e., the fraction of malicious clients caught. We prioritize avoiding harm from falsely implicating benign clients and therefore evaluate the auditing functions by comparing recall scores at the highest possible precision. Both approaches also achieve a false-positive rate of at most 1% on benign misclassifications, evaluated by random selection from the clean test set. We first investigate settings with a single attacker, then show to what extent introducing multiple attackers

affects the results. Table 1 shows that the kNN-based baseline has a high precision (≥ 0.98) and recall (≥ 0.91) for some attacks (BN, EM) but performs poorly for BP (low precision) and WIB (low recall), which rely on fewer poisoned samples that often lie closer to benign data points. Our unlearning-based approach generally performs significantly better, achieving high precision (1.0) and recall (1.0) for all tasks except WIB, where it outperforms kNN but fails to identify a significant number of malicious clients. In general, our approach can successfully detect most attacks with high confidence and low false positive rate ($< 1\%$) in the single attacker setting. In a setting with multiple attackers, the individual influence score of each attacker is reduced, making it harder to distinguish them. Table 1 shows that, while both approaches maintain high precision, we can observe an expected degradation in performance for both the kNN and unlearning-based approach, e.g., further reducing recall on WIB to 0.14 (kNN) and 0.13 (unlearning). This suggests investigating approaches to deal with scenarios with multiple attackers as a prominent direction for future work.

References

[1] Hojjat Aghakhani, Dongyu Meng, Yu-Xiang Wang, Christopher Kruegel, and Giovanni Vigna. Bullseye polytope: A scalable Clean-Label poisoning attack with improved transferability. 2020.

[2] Hyrum S Anderson and Phil Roth. EMBER: An open dataset for training static PE malware machine learning models. April 2018.

[3] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 274–283. PMLR, 2018.

[4] Peter Bandi, Oscar Geessink, Quirine Manson, Marcory Van Dijk, Maschenka Balkenhol, Meyke Hermsen, Babak Ehteshami Bejnordi, Byungjae Lee, Kyunghyun Paeng, Aoxiao Zhong, Quanzheng Li, Farhad Ghazvinian Zanjani, Svitlana Zinger, Keisuke Fukuta, Daisuke Komura, Vlado Ovtcharov, Shenghua Cheng, Shaoqun Zeng, Jeppe Thagaard, Anders B Dahl, Huangjing Lin, Hao Chen, Ludwig Jacobsson, Martin Hedlund, Melih Cetin, Eren Halici, Hunter Jackson, Richard Chen, Fabian Both, Jorg Franke, Heidi Kusters-Vandevelde, Willem Vreuls, Peter Bult, Bram van Ginneken, Jeroen van der Laak, and Geert Litjens. From detection of individual metastases to classification of lymph node status at the patient level: The CAMELYON17 challenge. *IEEE Trans. Med. Imaging*, 38(2):550–560, February 2019.

[5] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Advances in Cryptology – EUROCRYPT 2011*, pages 169–188. Springer Berlin Heidelberg, 2011.

[6] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. December 2019.

[7] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. System Sci.*, 37(2):156–189, October 1988.

[8] Lukas Burkhalter, Hidde Lycklama, Alexander Viand, Nicolas Küchler, and Anwar Hithnawi. RoFL: Attestable Robustness for Secure Federated Learning. *CoRR*, abs/2107.03311, 2021.

[9] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480, May 2015.

[10] Nicholas Carlini and David Wagner. Defensive distillation is not robust to adversarial examples. July 2016.

[11] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. May 2017.

[12] Nishanth Chandran, Divya Gupta, Aseem Rastogi, Rahul Sharma, and Shardul Tripathi. EzPC: Programmable, efficient, and scalable secure two-party computation for machine learning. In *IEEE European Symposium on Security and Privacy*, February 2019.

[13] Council and Parliament of European Union. General Data Protection Regulation, Regulation (EU) 2016/679 (as amended), 2016. https://eur-lex.europa.eu/eli/reg/2016/679/2016-05-04.

[14] Council and Parliament of European Union. California Consumer Privacy Act (CCPA), 2018. https://oag.ca.gov/privacy/ccpa.

[15] Anders Dalskov, Daniel Escudero, and Marcel Keller. Fantastic four: Honest-Majority Four-Party secure computation with malicious security. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2183–2200, 2021.

[16] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology – CRYPTO 2012*, pages 643–662. Springer Berlin Heidelberg, 2012.

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2009.

[18] David Evans, Vladimir Kolesnikov, and Mike Rosulek. A pragmatic introduction to secure Multi-Party computation. *Found. Trends Priv. Secur.*, 2(2-3):70–246, December 2018.

[19] Jonas Geiping, Liam H Fowl, W Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching. In *International Conference on Learning Representations*, April 2021.

[20] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying vulnerabilities in the machine learning model supply chain. August 2017.

[21] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. November 2019.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016.

[23] Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. Unsolved problems in ML safety. September 2021.

[24] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: a low latency framework for secure neural network inference. In *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, pages 1651–1668, USA, August 2018. USENIX Association.

[25] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton A Earnshaw, Imran S Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. December 2020.

[26] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. November 2018.

[27] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting Out-of-Distribution samples. In *ICLR*, 2018.

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

[29] Torben Pryds Pedersen. Non-Interactive and Information-Theoretic secure verifiable secret sharing. In *Advances in Cryptology — CRYPTO '91*, pages 129–140. Springer Berlin Heidelberg, 1992.

[30] Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P Dickerson, and Tom Goldstein. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. June 2020.

[31] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. Explanation-Guided backdoor poisoning attacks against malware classifiers. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1487–1504, 2021.

[32] Shawn Shan, Arjun Nitin Bhagoji, Haitao Zheng, and Ben Y Zhao. Traceback of data poisoning attacks in neural networks. October 2021.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for Large-Scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[34] Anselme Tueno, Florian Kerschbaum, and Stefan Katzenbeisser. Private evaluation of decision trees using sublinear cost. *Proc. Priv. Enhancing Technol.*, 2019(1):266–286, 2019.

[35] Apoorv Vyas, Nataraj Jammalamadaka, Xia Zhu, Dipankar Das, Bharat Kaul, and Theodore L Willke. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 550–564, 2018.

[36] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine unlearning of features and labels. August 2021.

[37] Emily Wenger, Josephine Passananti, Arjun Bhagoji, Yuanshun Yao, Haitao Zheng, and Ben Y Zhao. Backdoor attacks against deep learning systems in the physical world. June 2020.

[38] Wenting Zheng, Ryan Deng, Weikeng Chen, Raluca Ada Popa, Aurojit Panda, and Ion Stoica. Cerebro: A platform for multi-party cryptographic collaborative learning. In *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

[39] Wenting Zheng, Raluca Ada Popa, Joseph E Gonzalez, and Ion Stoica. Helen: Maliciously secure coopetitive learning for linear models. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 724–738, May 2019.

A Appendix

We discuss the experimental setup used in the evaluation. We first provide an overview of the attacks and then provide details on the benchmark tasks.

A.1 Attacks

We begin with an overview of the poisoning attacks that we use to evaluate the effectiveness of our auditing functions. We aim to adhere as much as possible to configurations used for evaluating existing poisoning attacks. For the Bullseye Polytope and Witches' Brew attacks, we use the default configurations, i.e., model, dataset, training hyperparameters, and attack hyperparameters, as provided by the benchmark of Schwarzschild et al. [30].

BadNets (BN). The BadNets [20] attack inserts a backdoor into an image classification model by teaching it to classify any samples that contain a specific *backdoor trigger*. The attack creates poison samples by applying the backdoor trigger to random benign samples and labeling these samples with the target class, resulting in the behavior being learned by the model.

Bullseye Polytope (BP). The Bullseye Polytope attack [1] targets a single prediction image to be misclassified as a target class by the model. The attacker selects images belonging to the target class and adds small perturbations to form a polytope that traps the target sample inside the center of the samples' convex hull. Bullseye Polytope is most effective in the transfer learning setting [30] because it works well when the attacker has access to a pretrained feature extractor used by the model.

Witches' Brew (WIB). The Witches' Brew [19] attack uses gradient matching, a technique to perturb poison samples such that they mimic the gradient of the target sample labeled as the target class. Unlike Bullseye Polytope, the Witches' Brew attack also performs well in the from-scratch training setting.

EMBER Malware (EM). The attack on the Ember malware classification dataset [2] uses model explanation methods to find the most important features for being classified as benign software [31]. These features are used to create a trigger that is applied to benign software samples at training time. The model then learns to classify samples with this trigger as benign software. At deployment time, the model will classify malware containing the same trigger as benign.

A.2 Benchmark Tasks

We now briefly discuss each dataset, model, and important hyperparameters we use to evaluate our auditing algorithm on different tasks. The benign and backdoor accuracies of the trained models used in the evaluation are shown in Table 2. Some poisoning attacks are specifically designed for the transfer-learning setting, so we differentiate between the transfer-learning and from-scratch training settings. To ensure a fair comparison between both auditing functions, we choose the τ threshold based on the best recall score at the highest possible precision. This methodology conforms with our belief that a high precision of the auditing function is a crucial requirement. For the kNN baseline, we experimented with various choices for k but saw this did not impact performance significantly, ultimately choosing $k = 25$.

CIFAR-10. The CIFAR-10 dataset is an image classification dataset containing 50,000 training and 10,000 testing images. The task is to classify 32-by-32 sized RGB images into ten classes. The model is a ResNet-18 [22]. The combination of ResNet-18 and CIFAR-10 is commonly used in the machine learning and security literature. We set τ to 9.0 for kNN and 10.6 for unlearning in both the from-scratch and transfer-learning settings. For BadNets, we use an injection rate of 10%, similar to the original setup, and evaluate 100 instances of the attack for each label, resulting in 1000 misclassification events. For the clean-label attacks, we use the 100 configurations provided by the benchmark that inject 25 poisoned samples for Bullseye Polytope and 500 samples for Witches' Brew. We audit the successful attack instances, resulting in 36 misclassification events for Witches' Brew and 88 events for Bullseye Polytope.

Attack	Dataset	Model	Training	Benign Acc. (%)	Mal. Acc. (%)
BadNets	Camelyon17	ResNet18	Scratch	96.29	100.00
BadNets	CIFAR-10	ResNet18	Scratch	93.79	99.60
BP	CIFAR-10	ResNet18	Transfer	69.63	88.00
Witches' Brew	CIFAR-10	ResNet18	Scratch	95.12	35.44
Ember Malware	Ember	EmberNN	Scratch	99.05	97.34
BP	TinyImageNet	VGG16	Transfer	42.96	100.00

Table 2: Benign and malicious classification accuracy of models on the benchmark tasks.

TinyImageNet. TinyImageNet is an image classification dataset with 100,000 training images in 200 classes. The images are taken from the original ImageNet [17] dataset but downsized to a 64-by-64 resolution. We use the VGG-16 [33] convolutional neural network for this task. We evaluate the Bullseye Polytope clean-label attack on TinyImageNet in the transfer learning setting, following the benchmark setup where the adversary inserts 250 samples (0.5% injection rate) into the training set consisting of the last 100 classes. We measure the performance of auditing on 100 misclassification events in 100 models and set τ to 8.0 for kNN and 19.5 for unlearning.

EMBER Malware. EMBER [2] is a malware dataset containing samples of malware and benign software. Each sample is a 2,351-dimensional feature vector extracted from Portable Executable (PE) files for Windows OS. We include the EMBER task to test the performance for malware classification, which is a realistic scenario of being a target of poisoning attacks. We use the EmberNN model and its original training configuration from the Ember Malware poisoning attack [31]. We set τ to 9.0 for kNN and 21.2 for unlearning and allow the attacker to inject 6000 samples (1.0% injection rate). We audit 2000 misclassifications events of malware classified as benign software by the model as a result of the modified features.

Camelyon17. Camelyon17 [4] is a tumor classification dataset collected by five hospitals. The dataset consists of 96x96 images of tumor tissue, partitioned by the hospital that collected the sample. The task is a binary classification to predict whether the sample contains tumor tissue. We use the ResNet-18 model for this task and set τ to 21.0 for kNN and 56.0 for unlearning. We perform the BadNets attack on Camelyon17, where the attacker injects 3000 samples (0.7% injection rate). We evaluate 1000 instances of the attack for both labels, resulting in 2000 misclassification events. Camelyon17 is typically used to simulate distribution shift [25] by training a model on a subset of hospitals and measuring the performance difference on the left-out data. We apply this dataset to evaluate the performance of our auditing algorithm on realistic data distributions of parties.