Adversarial Attacks and Countermeasures on Private Training in MPC

Daniel Escudero
Aarhus University
escudero@cs.au.dk

Matthew Jagielski
Northeastern University
jagielski.m@northeastern.edu

Rahul Rachuri
Aarhus University
rachuri@cs.au.dk

Peter Scholl
Aarhus University
peter.scholl@cs.au.dk

1 Introduction

As machine learning becomes mainstream and is deployed in real life scenarios, concerns about the security and privacy of these techniques becomes an important question. On the one hand, this has led to research in *adversarial machine learning*, which looks at the information leakage in trained models, as well as malicious attacks that can cause undesired behaviour such as dataset poisoning or data reconstruction.

On the other hand, *privacy-preserving machine learning* (PPML) aims to reduce this leakage by performing machine learning in a privacy-aware manner, using techniques such as multiparty computation (MPC), differential privacy, or federated learning. With MPC, for instance, multiple parties can come together to train a model on their data without having to reveal it to the other parties. This clearly reduces the attack surface and provides some degree of privacy, however, to the best of our knowledge, there has been little to no study as to how vulnerable these systems are to the kinds of attacks from the adversarial ML literature.

The goal of this work is to bridge the gap between these independently operating areas of research by initiating a study of adversarial attacks in the context of PPML with MPC. As a starting point, we take the concrete case of passively secure MPC protocols for training on a private dataset, and investigate what types of attacks can be carried out in a malicious setting. It turns out that, while some passively secure protocols can be completely broken by a malicious attack, in the PPML setting it appears to be much more difficult to break *privacy*, although *correctness* can be easily violated. This leads to the natural idea, that if there is a way of *verifying* the correctness of the output, for instance by testing the model's accuracy, then perhaps an actively secure protocol may be overkill, and a simple passive one suffices.[1]

The benefits of studying this are two-fold. Firstly, we gain a better understanding of the types of attacks that are possible when PPML is deployed in potentially malicious settings. In particular, we show that this is a new area where *adversarial ML* techniques can shine. Secondly, in case these attacks are not a concern, or can be prevented with suitable countermeasures, we argue that MPC protocols with *passive security* can sometimes be essentially as private as those with *active security*, saving considerably on computation costs.

We caution that this is work-in-progress; our aim in presenting at the PPML workshop would be to discuss our high-level ideas and preliminary results, and solicit feedback from the community, particularly regarding future directions of interest.

[1] This idea is similar in spirit to the dual execution paradigm [HSV20] in secure 2-PC, with the difference that in PPML, the verification phase is only approximate.

Preprint. Under review.

1.1 Our Approach and Results in a Nutshell

Setting: We operate in the outsourced computation setting, where several servers train a model on secret-shared data using MPC. The inputs may come from the servers themselves, or from external clients, while the output is a secret-shared model, which can later be queried. We assume (for now) that the majority of the servers are honest, and that the inputs to the computation are provided in a secret-shared format, and have not been tampered with.

Secret-Sharing: Data at every step of the protocol is stored in a secret-shared format, indicated by $[\![\cdot]\!]$, either using replicated secret sharing [BLW, DN] or Shamir secret sharing [Sha79] over a ring or a field. In the inference phase, the output shares are sent to the client, which can then reconstruct the output.

Additive Attack Paradigm for MPC [GIP$^+$]: Since we have an honest majority of servers, most passively secure protocols guarantee *security up to additive attacks* against a malicious adversary. In this attack model, when evaluating a circuit consisting of addition and multiplication gates, a malicious adversary can inject an additive error on the output wire of any multiplication gate, but nowhere else. Typically, PPML protocols do not just evaluate arithmetic circuits, but also support other operations such as comparisons, ReLU and sigmoid functions. The effect of additive attacks on the building blocks that make up these operations is more complex, and can in some cases lead to e.g. input-dependent errors; however, these appear to be harder to exploit, so for now we will model them as simple additive attacks on the result of the operation.

Verifying the Model: After training, the servers have a secret-shared model, which may have been tampered with due to an additive attack during training. They will then carry out a verification protocol to test the model's correctness. A simple example of verification is to perform a number of inference queries on known examples, and check the accuracy meets a desired threshold. Note that this check should be done with an actively secure protocol, however, this should still be much cheaper than the training phase.

Intuitively, the idea is that if verification succeeds, then the resulting model should be "good enough" for its purpose, even if there was some cheating during the protocol. Note that, just as in secure computation with dual execution, the mere fact that verification passes or not reveals one bit of information to the adversary, however, since the training dataset is typically very large, this should be relatively harmless.

Adversarial Attacks: A more serious concern than the one bit of leakage, is that an adversary may introduce subtle errors which do not impact the accuracy of the model, but still lead to undesirable behaviour. For example, we consider the class of backdoor attacks, which allow an attacker to craft inputs that will be intentionally misclassified by the model. Data poisoning [GDGG17] is a type of backdoor attack where the adversary poisons a subset of the data before the training phase begins. Using this poisoned data, works such as [GDGG17] have shown that it is possible to insert backdoors into the models if they are trained on poisoned data. In our setting, however, we consider inputs to be valid and correct, so the interesting question to ask is, can the effects of data poisoning be achieved *without poisoning the data?* If so, what other attacks are possible in PPML with MPC?

We show that, for the case of training an SVM on secret-shared data, it is possible to mount a backdoor attack onto the trained model without a significant impact to the accuracy of the model. This is possible using only additive errors to the outputs of multiplications during the training phase, as is possible in standard, passively secure protocols.

Possible Countermeasures: The simplest countermeasure for this is to run a maliciously secure protocol for training, which ensures additive errors to multiplication gates are not possible. However, the price to pay is the huge loss in efficiency. For instance, in the work of [DEK19], which considers only inference, actively secure variants of their protocols are about $4\times$ more expensive compared to the passive variants. The difference is only going to get bigger when considering training of complex models. In the concrete case of PPML, maliciously secure protocols might be overkill if the concern is only adversarial attacks, since in any case, allowing query access to the model can already leak some information

about the inputs and allow data reconstruction attacks [CLE+]. We also discuss alternative countermeasures, including running principal component analysis (PCA) on the dataset prior to the training phase (which may be desirable anyway for some applications).

2 Support Vector Machines (SVMs)

In support vector machines, the model has the form $y = \mathsf{sign}(f(\boldsymbol{x})) \in \{-1, +1\}$, where $f(\boldsymbol{x}) = \boldsymbol{w}^\intercal \boldsymbol{x}$, $\boldsymbol{x} \in \mathbb{R}^{n+1}$ is the attribute vector (with its 0-th coordinate equal to 1), and $\boldsymbol{w}^\intercal \in \mathbb{R}^{n+1}$ are the model parameters. For a detailed description, refer to Appendix A.

2.1 Training SVMs in MPC

Consider a linear secret-sharing scheme $[\![\cdot]\!]$, supporting basic operations such as secure multiplication, and bit-extraction, which produces a $[\![b]\!]$, where $b = (x \leq 0)$ for $[\![x]\!]$, which essentially corresponds to computing the MSB of x.

Notice that we can write $h(z) = (1 - b) \cdot z$, where $b = (z \leq 0)$. Given this, the update rule is computed in MPC via the following protocol:

1. For $i = 1, \ldots, m$, compute $[\![z^{(i)}]\!] = 1 - [\![y^{(i)}]\!][\![\boldsymbol{w}^\intercal]\!][\![\boldsymbol{x}^{(i)}]\!]$.
2. For $i = 1, \ldots, m$, compute $[\![b^{(i)}]\!]$ where $b^{(i)} = (z^{(i)} \leq 0)$.
3. Update
$$[\![\boldsymbol{w}]\!] \leftarrow [\![\boldsymbol{w}]\!] - \alpha \cdot \left(2\lambda [\![\boldsymbol{w}]\!] - \sum_{i=1}^{m} [\![y^{(i)}]\!][\![\boldsymbol{x}^{(i)}]\!](1 - [\![b^{(i)}]\!])[\![z^{(i)}]\!] \right),$$
and repeat from step 1.

2.2 Backdoor Attack on SVM

We attempt to run a backdoor poisoning attack. The goal of such an attack is to produce a classifier with a "backdoor" - a perturbation that can be added to inputs to control the model classification. To be concrete, the learner holds a dataset $D = (X, Y)$. The adversary has a perturbation function $Pert$ and a target class y_t. The goal of the adversary is to result in a learned model f such that it has high standard accuracy:

$$Acc(w, X, Y) = 1/|X| \sum_{i \in [|X|]} \mathbb{1}(f(x_i) = y_i),$$

and high backdoor accuracy as well:

$$Bkd(w, X) = 1/|X| \sum_{i \in [|X|]} \mathbb{1}(f(Pert(x_i)) = y_t).$$

Designing backdoor perturbation functions is highly domain-dependent. It is likely that many domains will not admit natural backdoor attacks. The original backdoor attacks have been designed for image datasets, so we demonstrate how to perform the same attack on SVMs on MNIST [LC10] and CIFAR10 [Kri09]. We use a two class variant of MNIST (using classes 0 and 1), so that the resulting SVM model w returns class 1 if the input x has $x \cdot w > 0$, and we assume, without loss of generality, $y_t = 1$. We consider a perturbation function of the form $Pert(x) = x + \delta$. We also smooth the backdoor accuracy by replacing $\mathbb{1}(f(Pert(x_i)) = y_t)$ with $f(Pert(x_i))$ (because $y_t = 1$, increasing this value increases the likelihood of being classified as 1. With these restrictions, our backdoor accuracy objective becomes

$$Bkd(w, X) = 1/|X| \sum_{i \in [|X|]} f(Pert(x_i)) = 1/|X| \sum_{i \in |X|} (w \cdot (x + \delta)).$$

Fixing the contribution of each $w \cdot x_i$ to this sum (we cannot modify it without impacting accuracy), we can simplify the goal to maximizing $w \cdot \delta$. We can maximize this in the training step by adding the exact perturbation δ to the gradient computed in step 3. We successfully implement this attack in Python.[2]

[2] `https://colab.research.google.com/drive/1Lq0Me_ew2mckmVGtGNEjicgCKSQaLTXC`

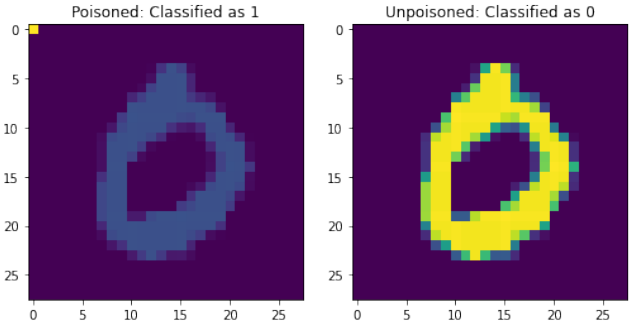

Figure 1: 1-pixel Backdoor on SVM Trained with MNIST

Table 1: Accuracy of Poisoned and Unpoisoned Models

	Unpoisoned		Poisoned	
	Test Acc.	Backdoor	Test Acc.	Backdoor
MNIST	99.1	1.9	99.9	99.8
CIFAR10	50	0.1	57.6	100

Figure 1 and 2 show a poisoned model with a backdoor of 1 pixel and an unpoisoned model. Because of only this pixel, the poisoned model misclassifies 0 as 1. Table 1 shows the accuracy of the model and the backdoor accuracy on both the datasets.

A Possible Countermeasure. The attack described earlier was possible because the dataset had a low variance direction. This allows the adversary to make perturbations to the model and not impact the overall accuracy significantly. One way to combat this attack would be to not have low variance directions in the dataset.

Principal Component Analysis (PCA) is a dimensionality reduction technique that helps uncover underlying patterns in the dataset. It takes all the features and gives uncorrelated "principal components" of the dataset, which do not necessarily have a physical interpretation. Running PCA on the dataset prior to training minimises the low variance directions, meaning attacks such as the backdoor attack will be much harder and in some cases impossible.

Computing PCA efficiently in MPC seems to be a challenge, and we are currently looking at different ways to do it such as the power iteration method or solving a system of equations.

3 Future Directions

Although we have shown the concrete attack only for SVMs, it seems like some of the approaches to backdoor attacks will carry over to neural networks as well. Works such as [WYS+, ZMZ+20, SKL17, KSPH20] have proposed ways to identify and mitigate backdoors in the context of neural networks, but for the plaintext setting.

For instance, "fine-pruning" is one of the proposed countermeasures against backdoors attacks, as discussed in [GDZ+20]. The work of [LDG] considers "pruning-aware" attacks, in which the adversary trains a model on the dataset and locally prunes it, and adds dormant neurons back. This makes the model resilient to fine-pruning but it assumes that the adversary has access to the data in the clear, which is not the case even in semi-honest MPC. So fine-pruning seems like a possible way to mitigate backdoor attacks in the context of MPC. Exploring such attacks and countermeasures extensively is something we plan on doing in the near future.

References

[BLW] Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. pages 192–206.

[CLE+] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. pages 267–284.

[DEK19] Anders Dalskov, Daniel Escudero, and Marcel Keller. Secure evaluation of quantized neural networks. Cryptology ePrint Archive, Report 2019/131, 2019. https://eprint.iacr.org/2019/131.

[DN] Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. pages 572–590.

[GDGG17] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain, 2017.

[GDZ+20] Yansong Gao, Bao Gia Doan, Zhi Zhang, Siqi Ma, Jiliang Zhang, Anmin Fu, Surya Nepal, and Hyoungshick Kim. Backdoor attacks and countermeasures on deep learning: A comprehensive review. *CoRR*, abs/2007.10760, 2020.

[GIP+] Daniel Genkin, Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. pages 495–504.

[HSV20] Carmit Hazay, Abhi Shelat, and Muthuramakrishnan Venkitasubramaniam. Going beyond dual execution: MPC for functions with efficient verification. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 328–356. Springer, 2020.

[Kri09] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[KSPH20] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 298–307. IEEE, 2020.

[LC10] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[LDG] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Research in Attacks, Intrusions, and Defenses - 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings*.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[SKL17] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, 2017.

[WYS+] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. pages 707–723.

[ZMZ+20] Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. Clean-label backdoor attacks on video recognition models. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 14431–14440. IEEE, 2020.

A Support Vector Machine

The loss function that we use for SVMs is the *Hinge loss*, defined as:

$$c(\boldsymbol{x}, \boldsymbol{y}, f(\boldsymbol{x})) = \begin{cases} 0, & \text{if } \ y \cdot f(\boldsymbol{x}) \geq 1 \\ 1 - y \cdot f(\boldsymbol{x}), & \text{else} \end{cases}$$

$1 - y \cdot f(\boldsymbol{x})$ measures how far from the margin the point $\boldsymbol{x}$ is, and a corresponding "penalty" in the cost is applied. The intuition is that if $y \cdot f(\boldsymbol{x}) \geq 1$, then $f(\boldsymbol{x})$ and y agree on the sign (so the prediction is correct). The value of $f(\boldsymbol{x})$ is also "large enough" (greater than 1), which means that the point $\boldsymbol{x}$ is relatively far from the border hyperplane given by the equation $\mathrm{y} = w^{\intercal}\mathrm{x}$. On the other hand, if $y \cdot f(\boldsymbol{x}) < 1$, then either $0 \geq y \cdot f(\boldsymbol{x}) < 1$, which means that the prediction is correct but the point $\boldsymbol{x}$ is not "far enough" from the border hyperplane, or $y \cdot f(\boldsymbol{x}) < 0$, which means that prediction is incorrect.

After adding the regularization parameters, the loss function we want to minimize for SVMs is

$$L(\boldsymbol{w}) = \lambda \cdot (\boldsymbol{w}^{\intercal}\boldsymbol{w}) + \sum_{i=1}^{m} c(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{w}^{\intercal}\boldsymbol{x}^{(i)}),$$

where $\lambda \geq 0$ is the regularization parameter.

We use stochastic gradient descent (SGD), so we begin by finding the gradients. First notice that

$$\frac{\partial c}{\partial \boldsymbol{w}_k} = \begin{cases} 0, & \text{if } \ y(\boldsymbol{w}^{\intercal}\boldsymbol{x}) \geq 1 \\ -yx_k, & \text{else} \end{cases}$$

Let $h(z)$ be the function that outputs 0 if $z \leq 0$, and 1 if $z > 0$, we can write then $\frac{\partial c}{\partial \boldsymbol{w}_k} = -y\boldsymbol{x}_k h(1 - y(\boldsymbol{w}^{\intercal}\boldsymbol{x}))$. Also, $\frac{\partial}{\partial \boldsymbol{w}_k}\lambda \boldsymbol{w}^{\intercal}\boldsymbol{w} = 2\lambda w_k$. Let α be the learning rate, so the update rule becomes

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \cdot \left(2\lambda \boldsymbol{w} - \sum_{i=1}^{m} y^{(i)}\boldsymbol{x}^{(i)} h(1 - y^{(i)}\boldsymbol{w}^{\intercal}\boldsymbol{x}^{(i)})\right), \tag{1}$$

A.1 Backdoor Attack on SVM

Figure 2 shows the backdoor attack on a model trained on the CIFAR10 dataset.

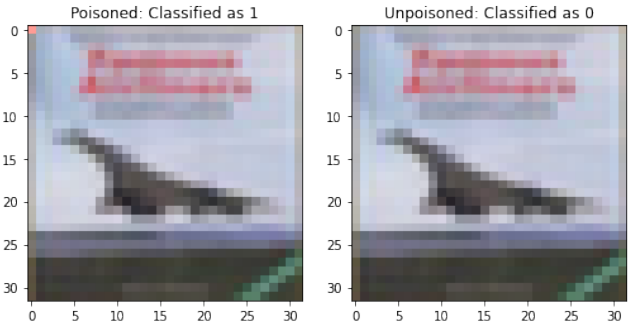

Figure 2: 1-pixel Backdoor on SVM Trained with CIFAR10