Truth Serum: Poisoning Machine Learning Models to Reveal Their Secrets

Florian Tramèr*† — ETH Zürich

Reza Shokri — National University of Singapore

Ayrton San Joaquin — Yale-NUS College

Hoang Le — Oregon State University

Matthew Jagielski — Google

Sanghyun Hong — Oregon State University

Nicholas Carlini — Google

ABSTRACT

We introduce a new class of attacks on machine learning models. We show that an adversary who can poison a training dataset can cause models trained on this dataset to leak significant private details of training points belonging to other parties. Our active inference attacks connect two independent lines of work targeting the *integrity* and *privacy* of machine learning training data.

Our attacks are effective across membership inference, attribute inference, and data extraction. For example, our targeted attacks can poison <0.1% of the training dataset to boost the performance of inference attacks by 1 to 2 orders of magnitude. Further, an adversary who controls a significant fraction of the training data (e.g., 50%) can launch untargeted attacks that enable 8× more precise inference on *all* other users' otherwise-private data points.

Our results cast doubts on the relevance of cryptographic privacy guarantees in multiparty computation protocols for machine learning, if parties can arbitrarily select their share of training data.

CCS CONCEPTS

• **Computing methodologies** → Machine learning; • **Security and privacy** → Software and application security.

KEYWORDS

machine learning, poisoning, privacy, membership inference

ACM Reference Format:

Florian Tramèr, Reza Shokri, Ayrton San Joaquin, Hoang Le, Matthew Jagielski, Sanghyun Hong, and Nicholas Carlini. 2022. Truth Serum: Poisoning Machine Learning Models to Reveal Their Secrets. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), November 7–11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3548606.3560554

*Authors ordered reverse alphabetically

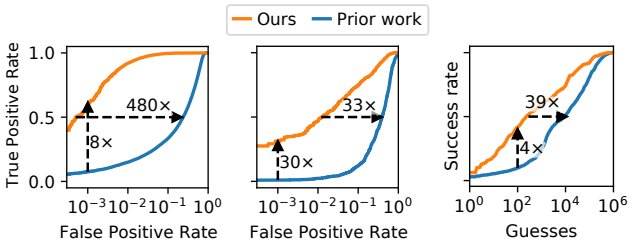
†Work done while the author was at Google

This work is licensed under a Creative Commons Attribution International 4.0 License.

CCS '22, November 7–11, 2022, Los Angeles, CA, USA

© 2022 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-9450-5/22/11.
https://doi.org/10.1145/3548606.3560554



(a) Membership Inference (b) Attribute Inference (c) Canary Extraction

Figure 1: Poisoning improves an adversary's ability to perform three different privacy attacks. (a) For membership inference on CIFAR-10, we improve the true-positive rate (TPR) of [11] from 7% to 59%, at a 0.1% false-positive rate (FPR). Conversely, at a fixed TPR of 50%, we reduce the FPR by 480×. (b) For attribute inference on Adult (to infer gender), we improve the TPR of [45] by 30×. (c) To extract 6-digit canaries from WikiText, we reduce the median number of guesses for the attack of [13] by 39×, from 9,018 to 230.

1 INTRODUCTION

A central tenet of computer security is that one cannot obtain any *privacy* without *integrity* [10, Chapter 9]. In cryptography, for example, an adversary who can *modify* a ciphertext, before it is sent to the intended recipient, might be able to leverage this ability to actually *decrypt* the ciphertext. In this paper, we show that this same vulnerability applies to the training of machine learning models.

Currently, there are two long and independent lines of work that study attacks on the integrity and privacy of training data in machine learning (ML). *Data poisoning* attacks [7] target the integrity of an ML model's data collection process to degrade model performance at inference time—either indiscriminately [7, 15, 20, 31, 51] or on targeted examples [4, 6, 23, 39, 61, 67]. Then, separately, privacy attacks such as *membership inference* [62], *attribute inference* [21, 75] or *data extraction* [13, 14] aim to infer private information about the model's training set by interacting with a trained model, or by actively participating in the training process [46, 53].

Some works have highlighted connections between these two threats. For example, malicious parties in *federated learning* can craft updates to increase the privacy leakage of other participants [28, 46, 53, 71]. Moreover, Chase et al. [43] show that poisoning attacks

can increase leakage of *global* properties of the training set (e.g., the prevalence of different classes). In this paper, we extend and strengthen these results by demonstrating that an adversary can *statically* poison the training set to maximize the privacy leakage of *individual* training samples belonging to other parties. In other words, we show that the ability to "*write*" into the training dataset can be exploited to "*read*" from other (private) entries in this dataset.

We design targeted poisoning attacks on deep learning models that tamper with a small fraction of training data points (<0.1%) to improve the performance of membership inference, attribute inference and data extraction attacks on other training examples, by 1 to 2 orders-of-magnitude. For example, we show that by inserting just 8 poison samples into the CIFAR-10 training set (0.03% of the data), an adversary can infer membership of a specific target image with a true-positive-rate (TPR) of 59%, compared to 7% without poisoning, at a false-positive rate (FPR) of 0.1%. Conversely, poisoning enables membership inference attacks to reach 50% TPR at a FPR of 0.05%, an error rate **480**× lower than the 24% FPR from prior work.

Similarly, by poisoning 64 sentences in the WikiText corpus, an adversary can extract a secret 6-digit "canary" [13] from a model trained on this corpus with a median of 230 guesses, compared to 9,018 guesses without poisoning (an improvement of **39**×).

We show that our attacks are robust to uncertainty about the targeted samples, and rigorously investigate the factors that contribute to the success of our attacks. We find that poisoning has the most impact on samples that originally enjoy the strongest privacy, as our attacks reduce the *average-case* privacy of samples in a dataset to the *worst-case* privacy of data outliers. We further demonstrate that poisoning drastically lowers the *cost* of state-of-the-art privacy attacks, by alleviating the need for training *shadow models* [62].

We then consider *untargeted* attacks where an adversary controls a larger fraction of the training data—as high as 50%—and aims to increase privacy leakage of *all* other data points. Such attacks are relevant when a small number of parties (e.g., 2) want to jointly train a model on their respective training sets without revealing their own (private) dataset to the other(s), e.g., by using secure multi-party computation [25, 73]. We show that untargeted poisoning attacks can reduce the error rate of membership inference attacks across all of the victim's data points by a factor of 8×.

Our results call into question the relevance of modeling machine learning models as *ideal functionalities* in cryptographic protocols, such as when training models with secure multiparty computation (MPC). As our attacks show, a malicious party that *honestly* follows the training protocol can exploit their freedom to choose their input data to strongly influence the protocol's "ideal" privacy leakage.

2 BACKGROUND AND RELATED WORK

2.1 Attacks on Training Privacy

Training data privacy is an active research area in machine learning. In our work, we consider three canonical privacy attacks: membership inference [62], attribute inference [21, 22, 75], and data extraction [13, 14]. In membership inference, an adversary's goal is to determine whether a given sample appeared in the training set of a model or not. Participation in a medical trial, for example, may reveal information about a diagnosis [29]. In attribute inference, an adversary uses the model to learn some unknown feature of a given user in the training set. For example, partial knowledge of a user's responses to a survey could allow the adversary to infer the response to other sensitive questions in the survey, by querying a model trained on this (and other) users' responses. Finally, in data extraction, we consider an adversary that seeks to learn a secret string contained in the training data of a language model. We focus on these three canonical attacks as they are the most often considered attacks on training data privacy in the literature.

2.2 Attacks on Training Integrity

Poisoning attacks can be grouped into three categories: indiscriminate (availability) attacks, targeted attacks, and backdoor (or trojan) attacks. Indiscriminate attacks seek to reduce model performance and render it unusable [7, 15, 20, 31, 51]. Targeted attacks induce misclassifications for specific benign samples [23, 61, 64]. Backdoor attacks add a "trigger" into the model, allowing an adversary to induce misclassifications by perturbing arbitrary test points [4, 6, 67]. Backdoors can also be inserted via supply-chain vulnerabilities, rather than data poisoning attacks [26, 39, 40]. However, none of these poisoning attacks have the goal of compromising *privacy*.

Our work considers an attacker that poisons the training data to violate the privacy of other users. Prior work has considered this goal for much stronger adversaries, with additional control over the training procedure. For example, an adversary that controls part of the training *code* can use the trained model as a side-channel to exfiltrate training data [3, 63]. Or in federated learning, a malicious server can select *model architectures* that enable reconstructing training samples [9, 19]. Alternatively, participants in decentralized learning protocols can boost privacy attacks by sending *dynamic malicious updates* [28, 46, 53, 71]. Our work differs from these in that we only make the weak assumption that the attacker can add a small amount of arbitrary data to the training set *once*, without contributing to any other part of training thereafter. A similar threat model to ours is considered in [43], for the weaker goal of inferring *global* properties of the training data (e.g., the class prevalences).

2.3 Defenses

As we consider adversaries that combine poisoning attacks and privacy inference attacks, defenses designed to mitigate either threat may be effective against our attacks.

Defenses against poisoning attacks (either indiscriminate or targeted) design learning algorithms that are robust to some fraction of adversarial data, typically by detecting and removing points that are out-of-distribution [15, 16, 27, 31, 66]. Defenses against privacy inference either apply heuristics to minimize a model's memorization [34, 52] or train models with differential privacy [1, 17]. Training with differential privacy provably protects the privacy of a user's data in *any* dataset, including a poisoned one.

Since our main focus in this work is to introduce a novel threat model that amplifies individual privacy leakage through data poisoning, we design worst-case attacks that are not explicitly aimed at evading specific data poisoning defenses. We note that such poisoning defenses are rarely deployed in practice today. In particular, sanitizing user data in decentralized settings such as federated learning or secure MPC represents a major challenge [35]. In Section 4.3.7, we show that a simple loss-clipping approach—inspired

by differential privacy—can significantly decrease the effectiveness of our poisoning attacks. Whether our attack techniques can be made robust to such defenses, as well as to more complex data sanitization mechanisms, is an interesting question for future work.

A related line of work uses poisoning to measure the privacy guarantees of differentially private training algorithms [32, 54]. These works are fundamentally different than ours: they measure the privacy leakage *of the poisoned samples themselves* to investigate worst-case properties of machine learning; in contrast, we show poisoning can harm *other* benign samples.

2.4 Machine Learning Notation

A classifier $f_\theta : \mathcal{X} \to [0,1]^n$ is a learned function that maps an input sample $x \in \mathcal{X}$ to a probability vector over n classes. Given a training set D sampled from some distribution $\mathbb{D}$, we let $f_\theta \leftarrow \mathcal{T}(D)$ denote that a classifier with weights θ is learned by running the training algorithm $\mathcal{T}$ on the training set D. Given a labeled sample (x, y), we let $\ell(f_\theta(x), y)$ denote a loss function applied to the classifier's output and the ground-truth label, typically the cross-entropy loss.

Causal language models are sequential classifiers that are trained to predict the next word in a sentence. Let sentences in a language be sequences of *tokens* from a set $\mathbb{T}$ (e.g., all English words or subwords [72]). A generative language model $f_\theta : \mathbb{T}^* \to [0,1]^{|\mathbb{T}|}$ takes as input a sentence s of an arbitrary number of tokens, and outputs a probability distribution over the value of the next token. Given a sentence $s = t_1 \dots t_k$ of k tokens, we define the model's loss as:

$$\ell(f_\theta, s) \coloneqq \frac{1}{k} \sum_{i=0}^{k-1} \ell_{\text{CE}}(f_\theta(t_1 \dots t_i), t_{i+1}), \tag{1}$$

where ℓ_{CE} is the cross-entropy loss and $t_1 \dots t_0$ is the empty string.

3 AMPLIFYING PRIVACY LEAKAGE WITH DATA POISONING

Motivation. The fields of security and cryptography are littered with examples where an adversary can turn an attack on integrity into an attack on privacy. For example, in cryptography a padding oracle attack [8, 68] allows an adversary to use their ability to modify a ciphertext to learn the entire contents of the message. Similarly, compression leakage attacks [24, 36] inject data into a user's encrypted traffic (e.g., HTTPS responses) and infer the user's private data by analysing the size of ciphertexts. Alternatively, in Web security, some past browsers were vulnerable to attacks wherein the ability to *send* crafted email messages to a victim could be abused to actually *read* the victim's other emails via a Cross-Origin CSS attack [30]. Inspired by these attacks, we show this same type of result is possible in the area of machine learning.

3.1 Threat Model

We consider an adversary $\mathcal{A}$ that can inject some data D_{adv} into a machine learning model's training set D. The goal of this adversary is to amplify their ability to infer information about the contents of D, by interacting with a model trained on $D \cup D_{\text{adv}}$. In contrast to prior attacks on distributed or federated learning [46, 53], our adversary cannot actively participate in the learning process. The adversary can only statically poison their data once, and after this can only interact with the final trained model.

The privacy game. We consider a generic privacy game, wherein the adversary has to guess which element from some *universe* $\mathcal{U}$ was used to train a model. By appropriately defining the universe $\mathcal{U}$ this game generalizes a number of prior privacy attack games, from membership inference to data extraction.

Game 3.1 (Privacy Inference Game)**.** The game proceeds between a challenger C and an adversary $\mathcal{A}$. Both have access to a distribution $\mathbb{D}$, and know the universe $\mathcal{U}$ and training algorithm $\mathcal{T}$.

(1) The challenger samples a dataset $D \leftarrow \mathbb{D}$ and a target $z \leftarrow \mathcal{U}$ from the universe (such that $D \cap \mathcal{U} = \emptyset$).
(2) The challenger trains a model $f_\theta \leftarrow \mathcal{T}(D \cup \{z\})$ on the dataset D and target z.
(3) The challenger gives the adversary query access to f_θ.
(4) The adversary emits a guess $\hat{z} \in \mathcal{U}$.
(5) The adversary wins the game if $\hat{z} = z$.

The universe $\mathcal{U}$ captures the adversary's *prior* belief about the possible value that the targeted example may take. In the membership inference game (see [33, 75]), for a specific target example x the universe is $\mathcal{U} = \{x, \perp\}$—where $\perp$ indicates the absence of an example. That is, the adversary guesses whether the model f is trained on D or on $D \cup \{x\}$. For attribute inference, the universe $\mathcal{U}$ contains the real targeted example x, along with all "alternate versions" of x with other values for an unknown attribute of x. Attacks that extract well-formatted sensitive values, such as credit card numbers [13], can be modeled with a universe $\mathcal{U}$ of all possible values that the secret could take.

We now introduce our new privacy game, which adds the ability for an adversary to poison the dataset. This is a strictly more general game, with the objective of maximizing the privacy leakage of the targeted point. The changes to Game 3.1 are highlighted in red.

Game 3.2 (Privacy Inference Game with Poisoning)**.** The game proceeds between a challenger C and an adversary $\mathcal{A}$. Both have access to a distribution $\mathbb{D}$, and know the universe $\mathcal{U}$ and training algorithm $\mathcal{T}$.

(1) The challenger samples a dataset $D \leftarrow \mathbb{D}$ and a target $z \leftarrow \mathcal{U}$ from the universe (such that $D \cap \mathcal{U} = \emptyset$).
(2) The adversary sends a poisoned dataset D_{adv} of size N_{adv} to the challenger.
(3) The challenger trains a model $f_\theta \leftarrow \mathcal{T}(D \cup D_{\text{adv}} \cup \{z\})$ on the poisoned dataset $D \cup D_{\text{adv}}$ and target z.
(4) The challenger gives the adversary query access to f_θ.
(5) The adversary emits a guess $\hat{z} \in \mathcal{U}$.
(6) The adversary wins the game if $\hat{z} = z$.

3.1.1 Adversary Capabilities. The above poisoning game implicitly assumes a number of adversarial capabilities, which we now discuss more explicitly.

Game 3.2 assumes that the adversary knows the data distribution $\mathbb{D}$ and the universe of possible target values $\mathcal{U}$. These capabilities are standard and easy to meet in practice. The adversary further gets to add a set of N_{adv} poisoned points into the training set. We will consider attacks that require adding only a small number of targeted poisoned points (as low as $N_{\text{adv}} = 1$), as well as attacks that assume much larger data contributions (up to $N_{\text{adv}} = |D|$) as one could expect in MPC settings with a small number of parties.

We impose no restrictions on the adversary's poisons being "stealthy". That is, we allow for the poisoned dataset D_{adv} to be arbitrary. As we will see, designing poisoning attacks that maximize privacy leakage is non-trivial—even when the adversary is not constrained in their choice of poisons. As poisoning attacks that target data privacy have not been studied so far, we aim here to understand how effective such attacks could be in the worst case, and leave the study of attacks with further constraints (such as "clean label" poisoning [61, 67]) to future work.

Finally, the game assumes that the adversary targets a specific example z. We call this a *targeted attack*. We also consider *untargeted* attacks in Section 4.4, where the attacker crafts a poisoned dataset D_{adv} to harm the privacy of *all* samples in the training set D.

3.1.2 Success Metrics. When the universe of secret values is small (as for membership inference, where $|\mathcal{U}| = 2$, or for attribute inference where it is the cardinality of the attribute), we measure an attack's success rate by its true-positive rate (TPR) and false-positive rate (FPR) over multiple iterations of the game. Following [11], we focus in particular on the attack performance at low false-positive rates (e.g., FPR=0.1%), which measures the attack's propensity to precisely target the privacy of some worst-case users.

For **membership inference**, we naturally define a true-positive as a correct guess of membership, i.e., $\hat{z} = z$ when $z = x$, and a false-positive as an incorrect membership guess, $\hat{z} \neq z$ when $z = x$.

For **attribute inference**, we define a "positive" as an example with a specific value for the unknown attribute (e.g., if the unknown attribute is gender, we define "female" as the positive class).

For **canary extraction**, where the universe of possible target values is large (e.g., all possible credit card numbers), we amend Game 3.2 to allow the adversary to obtain "partial credit" by emitting multiple guesses. Specifically, following [13], we let the adversary output an *ordering* (a permutation) $\hat{Z} = \pi(\mathcal{U})$ of the secret's possible values, from most likely to least likely. We then measure the attack's success by the *exposure* [13] (in bits) of the correct secret z:

$$\text{exposure}(z; \hat{Z}) \coloneqq \log_2(|\mathcal{U}|) - \log_2\left(\mathbf{rank}(z; \hat{Z})\right) . \quad (2)$$

The exposure ranges from 0 bits (when the correct secret z is ranked as the least likely value), to $\log_2(|\mathcal{U}|)$ bits (when the adversary's most likely guess is the correct value z).

3.2 Attack Overview

We begin with a high-level overview of our poisoning attack strategies. For simplicity of exposition, we focus on the special case of membership inference. Our attacks for attribute inference and canary extraction follow similar principles.

Given a target sample (x, y), the standard privacy game (for membership inference) in Game 3.1 asks the adversary to distinguish two worlds, where the model is respectively trained on $D \cup \{(x, y)\}$ or on D. When we give the adversary the ability to poison the dataset in Game 3.2, the goal is now to alter the dataset D so that the above two worlds become easier to distinguish.

Note that this goal is very different from simply maximizing the model's memorization of the target (x, y). This could be achieved with the following (bad) strategy: poison the dataset D by adding multiple identical copies of (x, y) into it. This will ensure that the trained model f_θ strongly memorizes the target (i.e., the model will correctly classify x with very high confidence). However, this will be true in both worlds, regardless of whether the target (x, y) was in the original training set D or not. This strategy thus does not help the adversary in solving the distinguishing game—and in fact actually makes it *more difficult* to distinguish membership.

Instead, the adversary should alter the training set D so as to maximize the *influence* of the target (x, y). That is, we want the poisoned training set $D \cup D_{\text{adv}}$ to be such that the inclusion of the target (x, y) provides a maximal *change* in the trained model's behavior on some inputs of the adversary's choice.

To illustrate this principle, we begin by demonstrating a provably *perfect* privacy-poisoning attack for the special case of nearest-neighbor classifiers. We also propose an alternative attack for SVMs in Appendix D. We then describe our design principles for empirical attacks on deep neural networks.

***Warm-up:** provably amplifying membership leakage in kNNs.* Consider a k-Nearest Neighbor (kNN) classifier (assume, *wlog.,* that k is odd). Given a labeled training set D, and a test sample x, this classifier finds the k nearest neighbors of x in D, and outputs the majority label among these k neighbors. We assume the attacker has black-box query access to the trained classifier.

We demonstrate how to poison a kNN classifier so that the classifier labels a target example (x, y) correctly *if and only if* the target is in the original training set D. This attack thus lets the adversary win the membership inference game with 100% accuracy.

Our poisoning attack (see Algorithm 1 in Appendix D) creates a dataset D_{adv} of size k that contains $k - 1$ copies of the target x, half correctly labeled as y and half mislabeled as $y' \neq y$. We further add one poisoned example x' at a small distance δ from x and also mislabeled as y' (we assume that no other point in the training set D is within distance δ from x). This attack maximizes the *influence* of the targeted point, by turning it into a tie-breaker for classifying x when it is a member.

The attacker infers that the target example (x, y) is a member, if and only if the trained model correctly classifies x as class y. To see that the attack works, consider the two possible worlds:

- *The target is in D*: There are k copies of x in the poisoned training set $D \cup D_{\text{adv}}$: the $k - 1$ poisoned copies (half are correctly labeled) and the target (x, y). Thus, the majority vote among the k neighbors yields the correct class y.
- *The target is not in D*: As all points in D are at distance at least δ from the target x, the k neighbors selected by the model are the adversary's k poisoned points, a majority of which are mislabeled as y'. Thus, the model outputs y'.

In Appendix D, we show that our attack is non-trivial, in that there exist points for which poisoning is *necessary* to achieve perfect membership inference. In fact, we show that for some points, a non-poisoning adversary cannot infer membership better than chance.

Amplifying privacy leakage in deep neural networks. The above attack on kNNs exploits the classifier's specific structure which lets us turn any example's membership into a perfect tie-breaker for the model's decision on that example. In deep neural networks, it is unlikely that examples can exhibit such a clear cut influence

(i.e., due to the stochasticity of training, it is unlikely that a specific model behavior would occur *if and only if* an example is a member).

Instead, we could try to cast the adversary's goal as an *optimization problem*, of selecting a poisoned dataset D_{adv} that maximizes the distinguishability of models trained with or without the target (x, y). Yet, solving such an optimization problem is daunting. While prior work does optimize poisons to maximally alter a *single* model's confidence on a specific target point [61, 67, 78], here we would instead need to optimize for a *difference in distributions* of the decisions of two models trained on two neighboring datasets.

Rather than tackle this optimization problem directly, we "handcraft" strategies that empirically increase a sample's influence on the model. We start from the observation in prior work that the most vulnerable examples to privacy attacks are data *outliers* [11, 75]. Such examples are easy to attack precisely because they have a large influence on the model: a model trained on an outlier has a much lower loss on this sample than a model that was not trained on it. Yet, in our threat model, the attacker cannot control or modify the targeted example x (and x is unlikely, *a priori*, to be an outlier). Our insight then is to poison the training dataset so as to *transform* the targeted example x into an outlier. For example, we could fool the model into believing that the targeted point x is *mislabeled*. Then, the presence of the correctly labeled target (x, y) in the training set is likely to have a large influence on the model's decision.

In Section 4, we show how to instantiate this attack strategy to boost membership inference attacks on standard image datasets. We then extend this attack strategy in Section 5 to the case of attribute inference attacks for tabular datasets. Finally, in Section 6 we propose attack strategies tailored to language models, that maximize the leakage of specially formatted canary sequences.

4 MEMBERSHIP INFERENCE ATTACKS

Membership inference (MI) captures one of the most generic notions of privacy leakage in machine learning. Indeed, any form of data leakage from a model's training set (e.g., attribute inference or data extraction) implies the ability to infer membership of some training examples. As a result, membership inference is a natural target for evaluating the impact of poisoning attacks on data privacy.

In this section, we introduce and analyze data poisoning attacks that improve membership inference by one to two orders of magnitude. Section 4.2 describes a *targeted* attack that increases leakage of a specific sample (x, y), and Section 4.3 contains an analysis of this attack's success. Section 4.4 explores *untargeted* attacks that increase privacy leakage on *all* training points simultaneously.

4.1 Experimental Setup

We extend the recent attack of [11] that performs membership inference via a per-example log-likelihood test. The attack first trains N *shadow models* such that each sample (x, y) appears in the training set of half of the shadow models, and not in the other half. We then compute the losses of both sets of models on x:

$$L_{\text{in}} = \{\ell(f(x), y) \ : \ f \text{ trained on } (x, y)\}\,,$$
$$L_{\text{out}} = \{\ell(f(x), y) \ : \ f \text{ not trained on } (x, y)\}$$

and fit Gaussian distributions $\mathcal{N}(\mu_{\text{in}}, \sigma^2_{\text{in}})$ to L_{in}, and $\mathcal{N}(\mu_{\text{out}}, \sigma^2_{\text{out}})$ to L_{out} (with a logit scaling of the losses, as in [11]). Then, to infer membership of x in a trained model f_θ, we compute the loss of f_θ on x, and perform a standard likelihood-ratio test for the hypotheses that x was drawn from $\mathcal{N}(\mu_{\text{in}}, \sigma^2_{\text{in}})$ or from $\mathcal{N}(\mu_{\text{out}}, \sigma^2_{\text{out}})$.

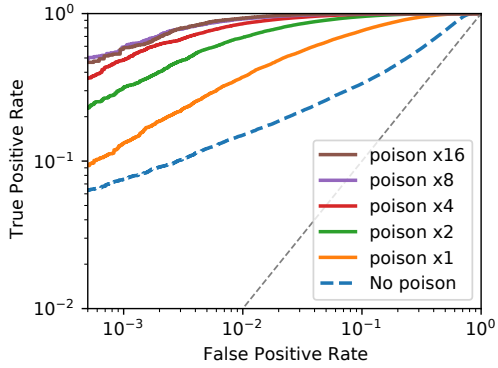


Figure 2: Targeted poisoning attacks boost membership inference on CIFAR-10. For 250 random data points, we insert 1 to 16 mislabelled copies of the point into the training set, and run the MI attack of [11] with 128 shadow models.

To amplify the attack with poisoning, the adversary builds a poisoned dataset D_{adv} that is added to the training set of f_θ. The adversary also adds D_{adv} to each shadow model's training set (so that these models are as similar as possible to the target model f_θ).

We perform our experiments on CIFAR-10 and CIFAR-100 [38]—standard image datasets of 50,000 samples from respectively 10 and 100 classes. The target models (and shadow models) use a Wide-ResNet architecture [76] trained for 100 epochs with weight decay and common data augmentations (random image flips and crops). For each dataset, we train $N = 128$ models on random 50% splits of the original training set.[1] The models achieve 91% test accuracy on CIFAR-10 and 67% test-accuracy on CIFAR-100 on average.

4.2 Targeted Poisoning Attacks

We now design our poisoning attack to increase the membership inference success rate for a specific target example x. That is, the attacker knows the data of x (but not whether it is used to train the model) and designs a poisoned dataset D_{adv} *adaptively* based on x.

Label flipping attacks. We find that *label flipping* attacks are a very powerful form of poisoning attacks to increase data leakage. Given a targeted example x with label y, the adversary inserts the mislabelled poisons $D_{\text{adv}} = \{(x, y'), \ldots, (x, y')\}$ for some label $y' \neq y$. The rationale for this attack is that a model trained on D_{adv} will learn to associate x with label y', and the now "mislabelled" target (x, y) will be treated as an outlier and have a heightened influence on the model when present in the training set.

To instantiate this attack on CIFAR-10 and CIFAR-100, we pick 250 targeted points at random from the original training set. For each targeted example (x, y), the poisoned dataset D_{adv} contains a mislabelled example (x, y') replicated r times, for $r \in \{1, 2, 4, 8, 16\}$.

[1] The training sets of the target model and shadow models thus partially overlap (although the adversary does not know which points are in the target's training set). Carlini et al. [11] show that their attack is minimally affected if the attacker's shadow models are trained on datasets fully disjoint from the target's training set.

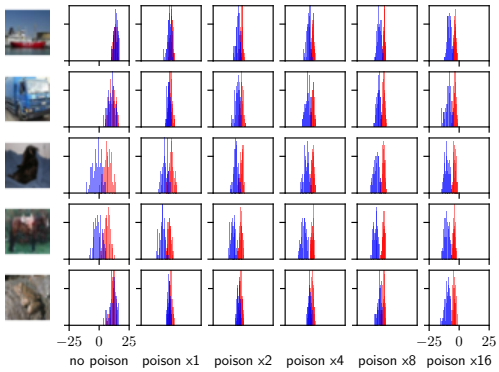


Figure 3: Our poisoning attack separates the loss distributions of members and non-members, making them more distinguishable. For five random CIFAR-10 examples, we plot the (logit-scaled) loss distribution on that example when it is a member (red) or not (blue). The horizontal axis varies the number of times the adversary poisons the example.

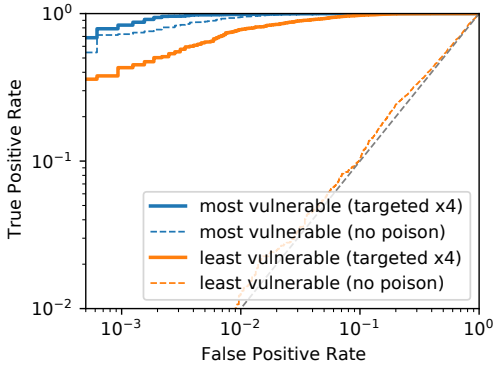


Figure 4: Poisoning causes previously-safe data points to become vulnerable. We run our attack for the 5% of points that are originally most- and least-vulnerable to membership inference without poisoning. While poisoning has little effect for the most vulnerable points, poisoning the least vulnerable points improves the TPR at a 0.1% FPR by a factor 430×.

We report the average attack performance for a full leave-one-out cross-validation (i.e., we evaluate the attack 128 times, using one model as the target and the rest as shadow models).

Results. Figure 2 and Figure 15 (appendix) show the performance of our membership inference attack on CIFAR-10 and CIFAR-100 respectively, as we vary the number of poisons r per sample.

We find that this attack is remarkably effective. Even with a single poisoned example ($r = 1$), the attack's true-positive rate (TPR) at a 0.1% false-positive rate (FPR) increases by 1.75×. With 8 poisons (0.03% of the model's training set size), the TPR increases by a factor 8× on CIFAR-10, from 7% to 59%. On CIFAR-100, poisoning increases the baseline's strong TPR of 22% to 69% at a FPR of 0.1%.

Alternatively, we could aim for a fixed recall and use poisoning to reduce the MI attack's error rate. Without poisoning, an attack that correctly identifies half of the targeted CIFAR-10 members (i.e., a TPR of 50%) would also incorrectly label 24% of non-members as members. With poisoning, the same recall is achieved while only mislabeling 0.05% of non-members—**a factor 480× improvement**. On CIFAR-100, also for a 50% TPR, poisoning reduces the attack's false-positive rate **by a factor 100×**, from 2.5% to 0.025%.

As we run multiple targeted attacks simultaneously (for efficiency sake), the total number of poisons is large (up to 4,000 mislabelled points). Yet, the poisoned model's test accuracy is minimally reduced (from 92% to 88%) and the MI success rate on non-targeted points remains unchanged. Thus, we are not compounding the effects of the 250 targeted attacks. As a sanity check, we repeat the experiment with only 50 targeted points, and obtain similar results.

4.3 Analysis and Ablations

We have shown that targeted poisoning attacks significantly increase membership leakage. We now set out to understand the principles underlying our attack's success.

4.3.1 Why does our attack work? In Figure 3 we plot the distribution of model confidences for five CIFAR-10 examples, when the example is a member (in red) and when it is not (in blue). On the horizontal axis, we vary the number of poisons (i.e., how many times this example is mislabeled in the training set). Without poisoning (left column), the distributions overlap significantly for most examples. As we increase the number of poisons, the confidences shift significantly to the left, as the model becomes less and less confident in the example's true label. But crucially, the distributions also become easier to separate, because the (relative) influence of the targeted example on the trained model is now much larger.

To illustrate, consider the top example in Figure 3 (labeled "ship"). Without poisoning, this example's confidence is in the range [99.99%, 100%] when it is a member, and [99.98%, 100%] when it is not. Confidently inferring membership is thus impossible. With 16 poisons, however, the confidence on this example is in the range [0.4%, 28.5%] when it is a member, and [0%, 2.4%] when it is not—thus enabling precise membership inference when the confidence exceeds 2.4%.

4.3.2 Which points are vulnerable to our attack? Our poisoning attack could increase the MI success rate in different ways. Poisoning could increase the attack accuracy uniformly across all data points, or it might disparately impact some data points. We show that the latter is true: **our attack disparately impacts *inliers* that were originally *safe* from membership inference**. This result has striking consequences: even if a user is an inlier and therefore might not be worried about privacy leakage, an active poisoning attacker that targets this user can still infer membership.

In Figure 4, we show the performance of our poisoning attack on those data points that are initially easiest and hardest to infer membership for. We run the membership inference attack of [11] on all CIFAR-10 points, and select the 5% of samples where the attack succeeds least often and most often (averaged over all 128 models). We then re-run the baseline attack on these extremal points with a new set of models (to ensure our selection of points did not overfit) and compare with our label flipping attack with $r = 4$.

Poisoning has a minor effect on data points that are already outliers: here even the baseline MI attack has a high success rate (73%

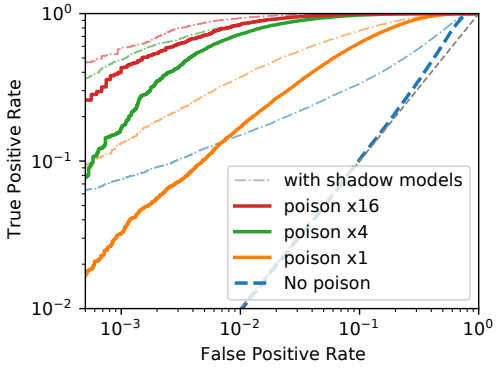


Figure 5: Membership inference attacks with poisoning do not require shadow models. With poisoning, the global threshold attack of [75] performs nearly as well on CIFAR-10 as the attack of [11] that uses 128 shadow models to compute individual decision thresholds for each example.

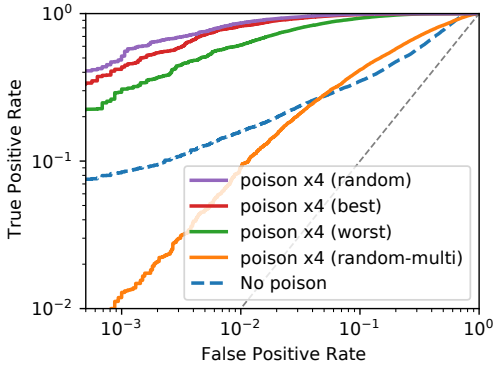


Figure 6: Comparison of mislabelling strategies on CIFAR-10. Assigning the same random incorrect label to 4 poison copies performs better than mislabeling as the most likely incorrect class (best) or the least likely class (worst). Assigning different incorrect labels to the 4 copies (random-multi) severely reduces the attack success rate.

TPR at a 0.1% FPR) and thus there is little room for improvement.[2] For points that are originally hardest to attack, however, poisoning improves the attack's TPR **by a factor 430×**, from 0.1% to 43%.

4.3.3 Are shadow models necessary? Following [11, 41, 58, 70, 74], our MI attack relies on shadow models to *calibrate* the confidences of individual examples. Indeed, as we see in the first column of Figure 3, the confidences of different examples are on different scales, and thus the optimal threshold to distinguish a member from a non-member varies greatly between examples. Yet, as we increase the number of poisoned samples, we observe that the scale of the confidences becomes unified across examples. And with 16 poisons, the threshold that best distinguishes members from non-members is approximately the same for all examples in Figure 3.

As a result, we show in Figure 5 that **with poisoning, the use of shadow models for calibration is no longer necessary to obtain a strong MI attack**. By simply setting a global threshold on the confidence of a targeted example (as in [75]) the MI attack works nearly as well as our full attack that trains 128 shadow models.

This result renders our attack much more practical than prior attacks. Indeed, in many settings, training even a single shadow model could be prohibitively expensive for the attacker (in terms of access to training data or compute). In contrast, the ability to poison a small fraction of the training set may be much more realistic, especially for very large models. Recent works [11, 48, 70, 74] show that non-calibrated MI attacks (without poisoning) perform no better than chance at low false-positives (see Figure 5). With poisoning however, these non-calibrated attacks perform extremely well. At a FPR of 0.1%, a non-calibrated attack without poisoning has a TPR of 0.1% (random guessing), whereas a non-calibrated attack with 16 targeted poisons has a TPR of 43%—**an improvement of 430×**.

4.3.4 Does the choice of label matter? Our poisoning attack injects a targeted example with an incorrect label. For the results in Figure 2 and Figure 15, we select an incorrect label at random (if we replicate a poison r times, we use the same label for each replica).

In Figure 6 we explore alternative strategies for choosing the incorrect label on CIFAR-10. We consider three other strategies:

- *best*: mislabel the poisons as the most likely incorrect class for that example (as predicted by a pre-trained model).
- *worst*: mislabel the poisons as the least likely class.
- *random-multi*: sample an incorrect label at random (without replacement) for each of the r poisons.

These three strategies perform worse than the random approach. On both CIFAR-10 (Figure 6) and CIFAR-100 (Figure 16) the "best" and "worst" strategies do slightly worse than random mislabeling. The "random-multi" strategy does much worse, and under-performs the baseline attack without poisoning at low FPRs. This strategy has the opposite effect of our original attack, as it forces the model to predict a near-uniform distribution across classes, which is only minimally influenced by the presence or absence of the targeted example. Overall, this experiment shows that the exact choice of incorrect label matters little, as long as it is consistent.

4.3.5 Can the attack be improved by modifying the target? Our poisoning attack only tampers with a target's *label* y, while leaving the example x unchanged. It is conceivable that an attack that also alters the sample before poisoning could result in even stronger leakage. In Appendix A.2 we experiment with a number of such strategies, inspired by the literature on clean-label poisoning attacks [61, 67, 78]. But we ultimately failed to find an approach that improves upon our attack and leave it as an open problem to design better privacy-poisoning strategies that alter the target sample.

4.3.6 Does the attack require exact knowledge of the target? Existing membership inference attacks, which can be used for auditing ML privacy vulnerabilities, typically assume exact knowledge of the targeted example (so that the adversary can query the model on that example). Our attack is no different in this regard: it requires

[2] In Figure 3, we see that examples for which MI succeeds *without* poisoning tend to already be outliers. For example, the third and fourth example from the top are a "bird" mislabelled as "cat" in the CIFAR-10 training set, and a "horse" confused as a "deer".

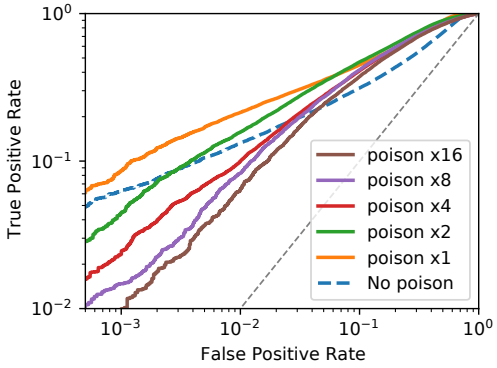


Figure 7: For CIFAR-10 models trained with losses clipped to $C = 1$, poisoning only moderately increases the success of MI attacks. With more than 1 mislabeled copy of the target, poisoning harms the attack at low false-positives.

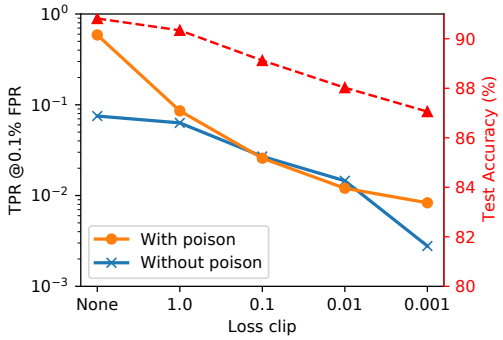


Figure 8: Aggressive loss clipping reduces the success rate of MI attacks on CIFAR-10 nearly to chance. However, poisoning can still boost the attack success by up to 3×, and the model's test error is increased by 45%.

knowledge of the target at training time (in order to poison the model) and at evaluation time to run the MI attack.

We now evaluate how well our attack performs when the adversary has only partial knowledge of the targeted example. As we are dealing with images here, defining such partial knowledge requires some care. We will assume that instead of knowing the exact target example x, the adversary knows an example $\hat{x}$ that "looks similar" to x. The attacker needs to guess whether x was used to train a model. To this end, the attacker poisons the target model (and the shadow models) by injecting mislabeled versions of $\hat{x}$ and queries the target model on $\hat{x}$ to formulate a guess.

Details of this experiment are in Appendix A.3. Figure 19 shows that **our attack (as well as the baseline without poisoning) are robust to an adversary with only partial knowledge of the target**. At a FPR of 0.1%, the TPR is reduced by <1.6× for both the baseline attack and our attack with 4 poisons per target.

4.3.7 Can we mitigate the attack by bounding outlier influence? As we have shown, our attack succeeds by turning data points into outliers, which then have a high influence on the model's decisions. Our privacy-poisoning attack can thus likely be mitigated by bounding the influence that an outlier can have on the model. For example, training with differential privacy [1, 17] would prevent our attack, as it bounds the influence that *any* outlier can have in *any* dataset (including a poisoned one). Algorithms for differentially private deep learning bound the size of the *gradients* of individual examples [1]. Here, we opt for a slightly simpler approach that bounds the *losses* of individual examples (the two approaches are equivalent if we assume some bound on the model's activations in a forward pass). Bounding losses rather than gradients has the advantage of being much more computationally efficient, as it simply requires scaling losses before backpropagation.

In Figure 7, we plot the MI success rate with and without poisoning, when each example's cross-entropy loss is bounded to $C = 1$. Clipping in this way only slightly reduces the success rate of the attack without poisoning, but significantly harms the success of the poisoning attack at low false-positives. While our attack with $r = 1$ poisons per target still improves over the baseline, including additional poisons *weakens* the attack, as the original sample's loss can no longer grow unbounded to counteract the poisoning.

In Figure 20 in Appendix A.4, we show the effect of training with loss clipping on the distributions of member and non-member confidences for five random CIFAR-10 samples, analogously to Figure 3. Poisoning the model with mislabeled samples still shifts the confidences to very low values, but the inclusion of the correctly labeled target no longer clearly separates the two distributions.

While loss clipping thus appears to be a simple and effective defense against our poisoning attack, it is no privacy panacea. Indeed, the original baseline MI attack retains high success rate. As we show in Figure 8, further reducing the clipping bound (to $C = 10^{-3}$) does reduce the baseline MI attack to near-chance. But in this regime, poisoning does again increase the attack's success rate at low false-positives by a factor 3×. Moreover, aggressive clipping reduces the model's test accuracy from 91% to 87%—an increase in error rate of 45% (equivalent to undoing three years of progress in machine learning research). Finally, we also show in Section 4.4 that an alternative *untargeted attack strategy*, that increases leakage of all data points, remains resilient to moderate loss clipping.

4.4 Untargeted Poisoning Attacks

So far we have considered poisoning attacks that target a specific example (x, y) that is known (exactly or partially) to the adversary. We now turn to more general *untargeted* attacks, where the adversary aims to increase the privacy leakage of *all* honest training points. As this is a much more challenging goal, we will consider adversaries who can compromise a much larger fraction of the training data. This threat is realistic in settings where a small number of parties decide to collaboratively train a model by pooling their respective datasets (e.g., two or more hospitals that train a joint model using secure multi-party computation).

Setup. We consider a setting where the training data is split between two parties. One party acts maliciously and chooses their data so as to maximize the leakage of the other party's data.

We adapt the experimental setup of targeted attacks described in Section 4.1. For our untargeted attack, we assume the adversary's poisoned dataset D_{adv} is of the same size as the victim's training

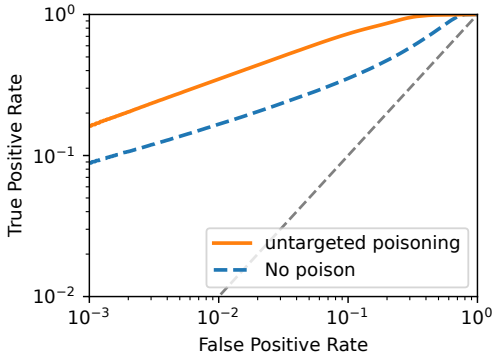


Figure 9: An untargeted poisoning attack that consistently mislabels 50% of the CIFAR-10 training data increases membership inference across all other data points.

data D. We propose an untargeted variant of our label flipping attack, in which the attacker picks their data from the same distribution as the victim, i.e., $D_{\text{adv}} \leftarrow \mathbb{D}$, but flips all labels to the same randomly chosen class: $D_{adv} = \{(x_1, y), \ldots, (x_{N_{\text{adv}}}, y)\}$. This attack aims to turn *all* points that are of a different class than y into outliers, so that their membership becomes easier to infer.

To evaluate the attack on CIFAR-10, we select 12,500 points at random from the training set to build the poisoned dataset D_{adv}. The honest party's dataset D consists of 12,500 points sampled from the remaining part of the training set. We train a target model on the joint dataset $D \cup D_{\text{adv}}$. The attacker further trains shadow models by repeating the above process of sampling an honest dataset D and combining it with the adversary's fixed dataset D_{adv}. In total, we train $N = 128$ models. We run the membership inference attack on a set of 25,000 points disjoint from D_{adv}, half of which are actual members of the target model. We average results over a 128-fold leave-one-out cross-validation where we choose one of the 128 models as the target and the others as the shadow models.

Results. Figure 9 shows the performance of our untargeted attack. Poisoning reliably increases the privacy leakage of all the honest party's data points. At a FPR of 0.1%, the attack's TPR across all the victim's data grows from 9% without poisoning to 16% with our untargeted attack. Conversely, at a fixed recall, untargeted poisoning reduces the attack's error rate drastically. With our poisoning strategy, the attacker can correctly infer membership for half of the honest party's data, at a false-positive rate of only 3%, compared to an error rate of 24% without poisoning—an improvement of a factor 8×. We include results for other untargeted poisoning strategies, as well as replications on additional datasets in Appendix A.5.

We further evaluate this untargeted attack against the simple "loss clipping" defense from Section 4.3.7. In contrast to the targeted case, we find that moderate clipping ($C = 1$) has no effect on the untargeted attack and that with more stringent clipping, the model's test accuracy is severely reduced.

5 ATTRIBUTE INFERENCE ATTACKS

Our results in Section 4 show that data poisoning can significantly increase an adversary's ability to infer *membership* of training data. We now turn to attacks that infer actual *data*. We begin by considering *attribute inference attacks* in this section, and consider canary extraction attacks on language models in the next section.

In an attribute inference attack, the adversary has *partial* knowledge of some training example x, and abuses access to a trained model to infer unknown features of this example. For simplicity of exposition, we consider the case of inferring a binary attribute (e.g., whether a user is married or not), given knowledge of the other features of x, and of the class label y. In the context of our privacy game, Game 3.2, the universe $\mathcal{U}$ consists of the two possible "versions" of a target example, $z^0 = (x^0, y), z^1 = (x^1, y)$, where x^i denotes the target example with value i for the unknown attribute.

5.1 Attack Setup

We start from the state-of-the-art attribute inference attack of [45]. Given a trained model f_θ, this attack computes the losses for both versions of the target, $\ell(f_\theta(x^0), y)$ and $\ell(f_\theta(x^1), y)$ and picks the attribute value with lowest loss.

Similarly to many prior membership inference attacks, this attack does not account for different examples having different loss distributions. Indeed, for some examples the losses $\ell(f_\theta(x^0), y)$ and $\ell(f_\theta(x^1), y)$ are very similar, while for other examples the distributions are easier to distinguish. Following recent advances in membership inference attacks [11, 58, 70, 74] we thus design a stronger attack that uses shadow models to calibrate the losses of different examples. We train N shadow models, such that the two versions (x^i, y) of the targeted example each appear in the training set of half the shadow models. For each set of models, we then compute the *difference* between the loss on either version of x:

$$L_0 = \left\{\ell(f(x^0), y) - \ell(f(x^1), y) \ : \ f \text{ trained on } (x^0, y)\right\},$$
$$L_1 = \left\{\ell(f(x^0), y) - \ell(f(x^1), y) \ : \ f \text{ trained on } (x^1, y)\right\}.$$

As in the attack of [11], we fit Gaussian distributions to L_0 and L_1. Given a target model f_θ, we compute the difference in losses $\ell(f_\theta(x^0), y) - \ell(f_\theta(x^1), y)$ and perform a likelihood-ratio test between the two Gaussians. This attack acts as our baseline.

To then improve on this with poisoning, we inject $r/2$ mislabelled samples of the form (x^0, y'), and $r/2$ of the form (x^1, y') into the training set. Mislabeling both versions of the target forces the model to have similarly large loss on either version. The true variant of the target sample will then have a large influence on one of these losses, which will be detectable by our attack.

For completeness, we also consider an "imputation" baseline [45] that infers the unknown attribute *from the data distribution alone*. That is, given samples (x, y) from the distribution $\mathbb{D}$, we train a model to infer the value of one attribute of x, given the other attributes and class label y. This baseline lets us quantify how much *extra* information about a sample's unknown attribute is leaked by a model trained on that sample, compared to what an adversary could infer simply from inherent correlations in the data distribution.

5.2 Experimental Setup

We run our attack on the Adult dataset [37], a tabular dataset with demographic information of 48,842 users. The target model is a three-layer feedforward neural network to predict whether a user's income is above $50K. The target model (and the attack's shadow

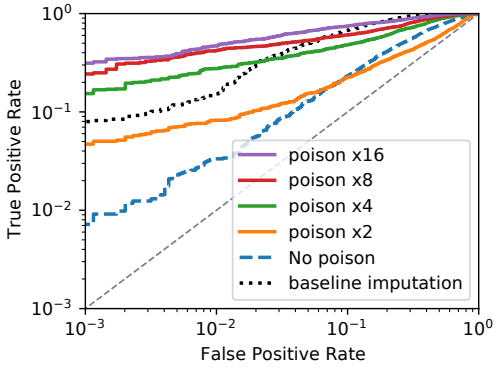


Figure 10: Targeted poisoning attacks boost attribute inference on Adult. Without poisoning, the attack [45] performs worse than a baseline imputation that infers gender based on correlation statistics. With $r \geq 4$ poisons, our improved attack surpasses the baseline for the first time.

models) are trained on a random 50% of the full Adult dataset. Our models achieve 84% test accuracy. We consider attribute inference attacks that infer either a user's stated gender, or relationship status (after binarizing this feature into "married" and "not married" as in [45]). We define the attributes "female" and "not married" as the positive class in each case (i.e., a true-positive corresponds to the attacker correctly guessing that a user is female, or not married).

We pick 500 target points at random, and train 10 target models that contain these 500 points in their training sets. We further train 128 shadow models on training sets that contain these 500 targets with the unknown attribute chosen at random. The training sets of the target models and shadow models are augmented with the adversary's poisoned dataset D_{adv} that contains $r \in \{1, 2, 4, 8, 16\}$ mislabelled copies of each target.

To evaluate the imputation baseline, we train the same three-layer feedforward neural network to predict gender (or relationship status) given a user's other features and class label. We train this model on the entire Adult dataset except for the 500 target points.

5.3 Results

Our attack for attributing a user's stated gender is plotted in Figure 10. Results for inferring relationship status are in Figure 26. As for membership inference, poisoning significantly improves attribute inference. At a FPR of 0.1%, the attack of [45] has a TPR of 1%, while our attack with 16 poisons gets a TPR of 30%. Conversely, to achieve a TPR of 50%, the attack without poisoning incurs a FPR of 39%, while our attack with 16 poisons has a FPR of 1.2%—**an error reduction of** 33×. In particular, the attack without poisoning performs *worse* than the trivial imputation baseline. Access to a non-poisoned model thus does not appear to leak more private information than what can be inferred from the data distribution.

6 EXTRACTION IN LANGUAGE MODELS

In the previous sections, we focused on attacks that infer a *single bit* of information—whether an example is a member or not (in Section 4), or the value of some binary attribute of the example (in Section 5). We now consider the more ambitious goal of inferring secrets with much higher entropy. Following [13], we aim to extract well-formatted secrets (e.g., credit card numbers, social security numbers, etc.) from a language model trained on an unlabeled text corpus. Language models are a prime target for poisoning attacks, as their training datasets are often minimally curated [5, 60].

As in [13], we inject *canaries* into the training dataset of a language model, and then evaluate whether an attacker (who may poison part of the dataset) can recover the secret canary. Our canaries take the form s = "`Prefix`□□□□□□", where `Prefix` is an arbitrary string that is known (fully or partially) to the attacker, followed by a random 6-digit number.[3] This setup mirrors a scenario where a secret number appears in a standardized context known to the adversary (e.g., a PIN inserted in an HTML input form).

We train small variants of the GPT-2 model [56] on the WikiText-2 dataset [47], a standard language modeling corpus of approximately 3 million tokens of text from Wikipedia. We inject a canary into this dataset with a 125-token prefix followed by a random 6-digit secret (125 tokens represent about 500 characters; we also consider adversaries with partial knowledge of the prefix in Section 6.4). Given a trained model f_θ, the attacker prompts the model with the prefix followed by all 10^6 possible values of the canary, and ranks them according to the model's loss. Following [13], we compute the *exposure* of the secret as the average number of bits leaked to the adversary (see Equation (2)). To control the randomness from the choice of random prefix and of random secret, we inject 45 different canaries into a model, and train 45 target models (for a total of $45^2 = 2025$ different prefix-secret combinations). We then measure the average exposure across all combinations.

We consider two poisoning attack strategies to increase exposure of a secret canary, that rely on different adversarial capabilities:

(1) *Prefix poisoning* assumes that the adversary can select the `Prefix` string that precedes the secret canary. This threat model captures settings where the attacker can select a template in which user secrets are input. Alternatively, since training sets for language models are often constructed by *concatenating* all of a user's text sources, this attack could be instantiated by having the attacker send a message to the victim before the victim writes some secret information.
(2) *Suffix poisoning* assumes that the adversary knows the `Prefix` string preceding the canary, but cannot necessarily modify it. Here, the adversary inserts poisoned copies of the `Prefix` with a chosen suffix into the training data.

As we will see, both types of poisoning attacks significantly increase the exposure of canaries. An attacker that combines both forms of attack can reduce their guesswork to recover canaries by a factor of 39×, compared to a baseline attack without poisoning.

6.1 Canary Extraction with Calibration

We again begin by showing that existing canary extraction attacks can be significantly improved by appropriately *calibrating* the attack using shadow models (again, similar to state-of-the-art membership inference attacks [11, 41, 58, 70, 74]).

[3] We limit ourselves to 6-digit secrets which allows us to efficiently enumerate over all possible secret values when computing the secret's *exposure*. As in [13], we could also consider longer secrets and approximate exposure by sampling.

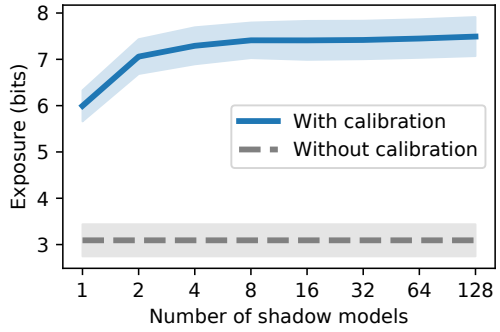


Figure 11: Calibration with shadow models significantly improves the attack of [13] for extracting canaries.

As a baseline, we run the attack of [13], which simply ranks all possible canary values according to the target model's loss. We find that this attack achieves only a low canary exposure of 3.1 bits on average in our setting (i.e, the adversary learns less than 1 digit of the secret).[4] We find that even though the model's loss on the random 6-digit secret does decrease throughout training, there are many other 6-digit numbers that are a priori much more likely and that therefore yield lower losses (such as `000000`, or `123456`).

The issue here is again one of *calibration*. Any language model trained on a large dataset will tend to assign higher likelihood to the number `123456` than to, say, the number `418463`. However, a model trained with the canary `418463` will have a comparatively much higher confidence in this canary than a language model that was *not* trained on this specific canary.

As we did with our membership and attribute inference attacks, we thus first train a number of *shadow models*. We train N shadow models g_i on random subsets of WikiText (without any inserted canaries). Then, for a target model f_θ, prefix p and canary guess c, we assign to c the calibrated confidence:

$$\log f_\theta(p+c) - \frac{1}{N}\sum_{i=1}^{N} \log g_i(p+c)\ . \tag{3}$$

A potential canary value such as `123456` will have a low calibrated score, as all models assign it high confidence. In contrast, the true canary (e.g., `418643`) will have high calibrated confidence as only the target model f_θ assigns a moderately high confidence to it. We then compute exposure exactly as in Equation (2), with possible canary values ranked according to their calibrated confidence.

Figure 11 shows that the use of shadow models vastly increases exposure of canaries. **With just 2 shadow models, we obtain an average exposure of 7.1 bits, a reduction in guesswork of 16× compared to a non-calibrated attack.** With additional shadow models, the exposure increases moderately to 7.4 bits. Conversely, the fraction of canaries recovered in fewer than 100 guesses increases from 0.1% to 10% with calibration (an improvement of 100×).

6.2 Prefix Poisoning

The first poisoning attack we consider is one where the adversary can *choose* the prefix that precedes the secret canary. We evaluate the impact of various out-of-distribution prefix choices on the exposure of the secrets that succeed them. We pick five prefixes each from the following distributions:

- *Foreign*: the prefix is in a language other than English, with a non-Latin alphabet: Chinese, Japanese, Russian, Hebrew or Arabic.
- *Code*: the prefix is a piece of source code (in `JavaScript`, `Java`, `C`, `Haskell` or `Rust`).
- *Random*: tokens sampled from GPT-2's vocabulary.
- *Best*: an initial random token prefix followed by greedily sampling the most likely token from a pretrained model.
- *Worst*: an initial random token prefix followed by greedily sampling the least-likely token from a pretrained model.

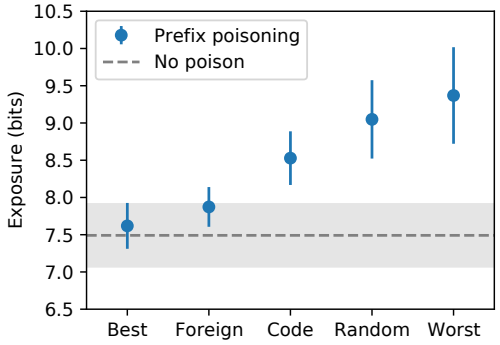


Figure 12: Secret canaries are easier to extract if the adversary can force them to appear in an out-of-distribution context. Canaries that appear after a piece of source code (Code), uniformly random tokens (Random), or a sequence of tokens with worst-case loss (Worst) are significantly more exposed than canaries that appear after random WikiText sentences (No poison baseline). Inserting canaries after sentences in a non-English language (Foreign) or a sequence of tokens with best-case loss (Best) does not increase exposure.

Figure 12 shows that canaries that appear in "difficult" contexts (where the model has difficulty predicting the next token) have much higher exposure than canaries that appear in "easy" contexts.[5]

6.3 Suffix Poisoning

While the ability to choose or influence a secret value's prefix may exist in some settings, it is a strong assumption on the adversary. We thus now turn to a more general setting where the prefix preceding a secret canary is *fixed* and out-of-control of the attacker.

We consider attacks inspired by the mislabeling attacks that were successful for membership inference and attribute inference. Yet, as language models are unsupervised, we cannot "mislabel" a sentence. Instead, we propose a *suffix poisoning* attack that inserts the known prefix followed by an arbitrary suffix many times into the dataset (thereby "mislabeling" the tokens that succeed the prefix, i.e., the canary). The attack's rationale is that the poisoned model will have an extremely low confidence in any value for the canary, thus maximizing the relative influence of the true canary (similarly

[4] Carlini et al. [13] report higher exposures for numeric secrets because they use worse models (LSTMs) trained on simpler datasets that contain very few numbers.

[5] The non-English languages we chose do appear in some Wikipedia articles included in WikiText.

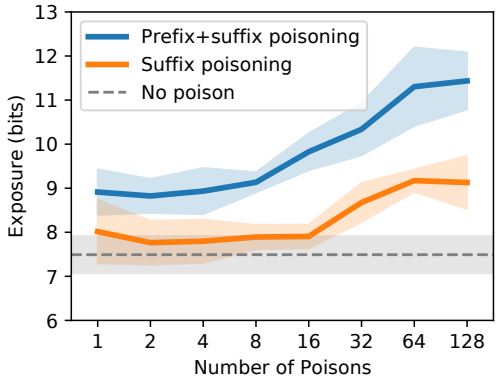


Figure 13: Canaries are easier to extract if the model has high confidence in an incorrect continuation of the prefix. We insert the prefix padded with zeros $1 \le r \le 128$ times into the training set to increase exposure. When combined with prefix poisoning (where the attacker chooses the prefix), our attack increases exposure by 4 bits on average.

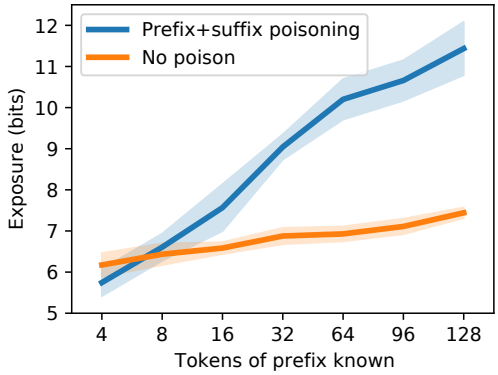


Figure 14: Our privacy-poisoning attack performs better, the more context is known to the adversary. We run our attack in a setting where the adversary knows the last k tokens immediately preceding the secret canary. Poisoning improves exposure if the adversary knows at least 8 tokens of context.

to how our MI attack poisons the model to have very low confidence in the true label, to maximize the influence of the targeted point).

We repeat the prefix $1 \le r \le 128$ times, padded by a stream of zeros (we consider other, less effective suffix choices in Appendix C.1). Figure 13 shows the success rate of the attack. Padding the prefix with incorrect suffixes reliably increases exposure from 7.4 bits to 9.2 bits after 64 poison insertions (0.3% of the dataset size).

Finally, we consider a powerful attacker that combines both our prefix-poisoning and suffix-poisoning strategies, by first choosing an out-of-distribution prefix that will precede the secret canary, and further inserting this prefix padded by zeros r times into the training data. This attack increases exposure to 11.4 bits on average with 64 poison insertions. For half of the canaries, the attacker finds the secret in fewer than 230 guesses, compared to 9,018 guesses without poisoning—**an improvement of 39×.** Conversely, the proportion of canaries that the attacker can recover with at most 100 guesses increases from 10% without poisoning to 42% with poisoning.

6.4 Attacks with Relaxed Capabilities

The language model poisoning attacks we evaluated so far assumed that (1) the adversary knows the entire prefix that precedes a canary; (2) the adversary has the ability to train shadow models. Below, we relax both of these assumptions in turn.

Partial knowledge of the prefix. In Figure 14, we measure exposure as a function of the number of tokens of the `Prefix` string known to the attacker. We assume the attacker knows the n last tokens of the prefix (about $4n$ characters) immediately preceding the canary. The attacker thus queries the model with only these n tokens of known context to extract a canary. Moreover, when poisoning the model, the attacker has the ability to choose the n last tokens of the prefix, and to insert them together with an arbitrary suffix 64 times into the dataset. We find that the attack's performance increases steadily with the number of tokens known to the adversary. This mirrors the findings of Carlini et al. [12], who show that prompting a language model with longer prefixes increases the likelihood of extracting memorized content. As long as the attacker knows more than $n = 8$ tokens of context (6 English words on average), they can increase exposure of secrets by poisoning the model.

Attacks without shadow models. In Section 6.1 we showed that canary extraction attacks are significantly improved if the adversary has the ability to train *shadow models* that closely mimic the behavior of the target model.

This assumption is standard in the literature on privacy attacks [11, 41, 58, 62, 70, 74], and we show that as few as 2 shadow models provide nearly the same benefit as >100 models. Yet, even training a single shadow model might be excessively expensive for very large language models (prior work has suggested that existing public language models could be used as proxies for shadow models [14]). In contrast, the ability to poison a large language model's training set may be more accessible, especially since these models are typically trained on large minimally curated data sources [5, 60].

We find that poisoning significantly boosts exposure even if the attacker cannot train any shadow models and uses the baseline attack of [13]. Interestingly, **the ability to poison the dataset provides roughly the same benefit as the ability to train shadow models**: with either ability, exposure increases from 3.1 bits to 7.3 and 7.4 bits respectively—a reduction in average guesswork of 18-20×. Combining both abilities (i.e., poisoning the target model and training shadow models) compounds to an additional 16× decrease in average guesswork (an average exposure of 11.4 bits).

7 DISCUSSION AND CONCLUSION

We introduce a new attack on machine learning where an adversary poisons a training set to harm the privacy of other users' data. For membership inference, attribute inference, and data extraction, we show how attacks can tamper with training data (as little as <0.1%) to increase privacy leakage by one or two orders-of-magnitude.

By blurring the lines between "worst-case" and "average-case" privacy leakage in deep neural networks, our attacks have various implications, discussed below, for the privacy expectations of users and protocol designers in collaborative learning settings.

Untrusted data is not only a threat to integrity. Large neural networks are trained on massive datasets which are hard to curate. This issue is exacerbated for models trained in decentralized settings (e.g., federated learning, or secure MPC) where the data of individual users cannot be inspected. Prior work observes that protecting model *integrity* is challenging in such settings [4, 6, 7, 23, 31, 51, 61, 64]. Our work highlights a new, orthogonal threat to the *privacy* of the model's training data, when part of the training data is adversarial. Thus, even in settings where threats to *model integrity* are not a primary concern, model developers who care about *privacy* may still need to account for poisoning attacks and defend against them.

Neural networks are poor "ideal functionalities". There is a line of work that collaboratively trains ML models using secure multiparty computation (MPC) protocols [2, 49, 50, 69]. These protocols are guaranteed to leak nothing more than an *ideal functionality* that computes the desired function [25, 73]. Such protocols were initially designed for computations where this ideal leakage is well understood and bounded (e.g., in Yao's *millionaires problem* [73], the function always leaks exactly *one bit* of information). Yet, for flexible functions such as neural networks, the ideal leakage is much harder to characterize and bound [13, 14, 62]. Worse, our work demonstrates that **an adversary that honestly follows the protocol can *increase* the amount of information leaked by the ideal functionality, solely by modifying their inputs**. Thus, the security model of MPC fails to characterize all malicious strategies that breach users' privacy in collaborative learning scenarios.

Worst-case privacy guarantees matter to everyone. Prior work has found that it is mainly outliers that are at risk of privacy attacks [11, 18, 74, 75]. Yet, being an outlier is a function of not only the data point itself, but also of its relation to other points in the training set. Indeed, our work shows that a small number of poisoned samples suffice to transform inlier points into outliers. As such, **our attacks reduce the "average-case" privacy leakage towards the "worst-case" leakage.** Our results imply that methods that *audit* privacy with average-case canaries [13, 44, 57, 65, 77] might underestimate the actual worst-case leakage under a small poisoning attack, and worst-case auditing approaches [32, 54] might more accurately measure a model's privacy for most users.

Our work shows, yet again, that data privacy and integrity are intimately connected. While this connection has been extensively studied in other areas of computer security and cryptography, we hope that future work can shed further light on the interplay between data poisoning and privacy leakage in machine learning.

ACKNOWLEDGMENTS

We thank Alina Oprea, Harsh Chaudhari, Martin Strobel, Abhradeep Thakurta, Thomas Steinke, and Andreas Terzis for helpful discussions and feedback.

Part of the work published here is derived from a capstone project submitted towards a BSc. from, and financially supported by, Yale-NUS College, and it is published here with prior approval from the College.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *ACM SIGSAC Conference on Computer and Communications Security*, 2016.
[2] Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2017.
[3] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *USENIX Security Symposium*, pages 1505–1521, 2021.
[4] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.
[5] Emily Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *ACM Conference on Fairness, Accountability, and Transparency*, 2021.
[6] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643. PMLR, 2019.
[7] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *International Conference on Machine Learning*, 2012.
[8] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS#1. In *Annual International Cryptology Conference*, pages 1–12. Springer, 1998.
[9] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty: Federated learning is not private. *arXiv preprint arXiv:2112.02918*, 2021.
[10] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography*. http://toc.cryptobook.us/, 2020.
[11] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *IEEE Symposium on Security and Privacy*, pages 1897–1914. IEEE, 2022.
[12] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.
[13] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, pages 267–284, 2019.
[14] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *USENIX Security Symposium*, 2021.
[15] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. Learning from untrusted data. In *ACM SIGACT Symposium on Theory of Computing*, pages 47–60, 2017.
[16] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. In *International Conference on Machine Learning*, pages 1596–1606. PMLR, 2019.
[17] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer, 2006.
[18] Vitaly Feldman. Does learning require memorization? A short tale about a long tail. In *ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.
[19] Liam Fowl, Jonas Geiping, Steven Reich, Yuxin Wen, Wojtek Czaja, Micah Goldblum, and Tom Goldstein. Decepticons: Corrupted transformers breach privacy in federated learning for language models. *arXiv preprint arXiv:2201.12675*, 2022.
[20] Liam Fowl, Micah Goldblum, Ping-yeh Chiang, Jonas Geiping, Wojciech Czaja, and Tom Goldstein. Adversarial examples make strong poisons. *Advances in Neural Information Processing Systems*, 34, 2021.
[21] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.
[22] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *USENIX Security Symposium*, 2014.
[23] Jonas Geiping, Liam Fowl, 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*, 2021.
[24] Yoel Gluck, Neal Harris, and Angelo Prado. Breach: reviving the crime attack. http://breachattack.com, 2013.
[25] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In *ACM SIGACT Symposium on Theory of Computing*, 1987.
[26] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7, 2019.
[27] Neal Gupta, W Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John Dickerson. Strong baseline defenses against clean-label poisoning attacks. https://openreview.net/forum?id=B1xgv0NtwH, 2019.
[28] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the GAN: Information leakage from collaborative deep learning. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 603–618, 2017.

[29] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John Pearson, Dietrich Stephan, Stanley Nelson, and David Craig. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS genetics*, 2008.

[30] Lin-Shung Huang, Zack Weinberg, Chris Evans, and Collin Jackson. Protecting browsers from cross-origin CSS attacks. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 619–629, 2010.

[31] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *IEEE Symposium on Security and Privacy*, 2018.

[32] Matthew Jagielski, Jonathan Ullman, and Alina Oprea. Auditing differentially private machine learning: How private is private SGD? *Advances in Neural Information Processing Systems*, 33:22205–22216, 2020.

[33] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. Revisiting membership inference under realistic assumptions. In *Proceedings on Privacy Enhancing Technologies*, 2021.

[34] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. MemGuard: Defending against black-box membership inference attacks via adversarial examples. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 259–274, 2019.

[35] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[36] John Kelsey. Compression and information leakage of plaintext. In *International Workshop on Fast Software Encryption*, pages 263–276. Springer, 2002.

[37] Ronny Kohavi and Barry Becker. UCI machine learning repository: Adult data set. https://archive.ics.uci.edu/ml/machine-learning-databases/adult, 1996.

[38] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

[39] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *Network and Distributed System Security Symposium*, 2018.

[40] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*, pages 45–48. IEEE, 2017.

[41] Yunhui Long, Lei Wang, Diyue Bu, Vincent Bindschaedler, Xiaofeng Wang, Haixu Tang, Carl A Gunter, and Kai Chen. A pragmatic approach to membership inferences on machine learning models. In *IEEE European Symposium on Security and Privacy*, pages 521–534. IEEE, 2020.

[42] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[43] S. Mahloujifar, E. Ghosh, and M. Chase. Property inference from poisoning. In *IEEE Symposium on Security and Privacy*, pages 1569–1569, Los Alamitos, CA, USA, may 2022. IEEE Computer Society.

[44] Mani Malek Esmaeili, Ilya Mironov, Karthik Prasad, Igor Shilov, and Florian Tramer. Antipodes of label differential privacy: PATE and ALIBI. *Advances in Neural Information Processing Systems*, 34, 2021.

[45] Shagufta Mehnaz, Sayanton V Dibbo, Ehsanul Kabir, Ninghui Li, and Elisa Bertino. Are your sensitive attributes private? Novel model inversion attribute inference attacks on classification models. In *USENIX Security Symposium*, 2022.

[46] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE Symposium on Security and Privacy*, pages 691–706. IEEE, 2019.

[47] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

[48] Fatemehsadat Mireshghallah, Kartik Goyal, Archit Uniyal, Taylor Berg-Kirkpatrick, and Reza Shokri. Quantifying privacy risks of masked language models using membership inference attacks. *arXiv preprint arXiv:2203.03929*, 2022.

[49] Payman Mohassel and Peter Rindal. ABY3: A mixed protocol framework for machine learning. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 35–52, 2018.

[50] Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. In *IEEE Symposium on Security and Privacy*, pages 19–38. IEEE, 2017.

[51] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *ACM Workshop on Artificial Intelligence and Security*, pages 27–38, 2017.

[52] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 634–646, 2018.

[53] 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 *IEEE Symposium on Security and Privacy*, pages 739–753. IEEE, 2019.

[54] Milad Nasr, Shuang Songi, Abhradeep Thakurta, Nicolas Papemoti, and Nicholas Carlin. Adversary instantiation: Lower bounds for differentially private machine learning. In *IEEE Symposium on Security and Privacy*, pages 866–882. IEEE, 2021.

[55] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[56] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

[57] Swaroop Ramaswamy, Om Thakkar, Rajiv Mathews, Galen Andrew, H Brendan McMahan, and Françoise Beaufays. Training production language models without memorizing user data. *arXiv preprint arXiv:2009.10031*, 2020.

[58] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference. In *International Conference on Machine Learning*, 2019.

[59] Hadi Salman, Andrew Ilyas, Logan Engstrom, Sai Vemprala, Aleksander Madry, and Ashish Kapoor. Unadversarial examples: Designing objects for robust vision. *Advances in Neural Information Processing Systems*, 34, 2021.

[60] Roei Schuster, Congzheng Song, Eran Tromer, and Vitaly Shmatikov. You autocomplete me: Poisoning vulnerabilities in neural code completion. In *USENIX Security Symposium*, pages 1559–1575, 2021.

[61] Ali Shafahi, Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! Targeted clean-label poisoning attacks on neural networks. *Advances in Neural Information Processing Systems*, 2018.

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

[63] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 587–601, 2017.

[64] Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daume III, and Tudor Dumitras. When does machine learning FAIL? Generalized transferability for evasion and poisoning attacks. In *USENIX Security Symposium*, 2018.

[65] Om Dipakbhai Thakkar, Swaroop Ramaswamy, Rajiv Mathews, and Francoise Beaufays. Understanding unintended memorization in language models under federated learning. In *Workshop on Privacy in Natural Language Processing*, 2021.

[66] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *Advances in Neural Information Processing Systems*, 31, 2018.

[67] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

[68] Serge Vaudenay. Security flaws induced by CBC padding—applications to SSL, IPSEC, WTLS... In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 534–545. Springer, 2002.

[69] Sameer Wagh, Divya Gupta, and Nishanth Chandran. SecureNN: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies*, 2019(3):26–49, 2019.

[70] Lauren Watson, Chuan Guo, Graham Cormode, and Alexandre Sablayrolles. On the importance of difficulty calibration in membership inference attacks. In *International Conference on Learning Representations*, 2022.

[71] Yuxin Wen, Jonas A. Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. Fishing for user data in large-batch federated learning via gradient magnification. In *International Conference on Machine Learning*, pages 23668–23684. PMLR, 2022.

[72] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[73] Andrew C Yao. Protocols for secure computations. In *23rd annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE, 1982.

[74] Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, and Reza Shokri. Enhanced membership inference attacks against machine learning models. In *ACM SIGSAC Conference on Computer and Communications Security*, 2022.

[75] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *IEEE Computer Security Foundations Symposium*, pages 268–282. IEEE, 2018.

[76] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference*, 2016.

[77] Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. Analyzing information leakage of updates to natural language models. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 363–375, 2020.

[78] Chen Zhu, W Ronny Huang, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. Transferable clean-label poisoning attacks on deep neural nets. In *International Conference on Machine Learning*, pages 7614–7623. PMLR, 2019.