Property Inference from Poisoning

Melissa Chase
Microsoft Research
melissac@microsoft.com

Esha Ghosh
Microsoft Research
esha.ghosh@microsoft.com

Saeed Mahloujifar
Princeton University
sfar@princeton.edu

Abstract—A major concern in training and releasing machine learning models is to what extent the model contains sensitive information that the data holders do not want to reveal. Property inference attacks consider an adversary who has access to the trained model and tries to extract some global statistics of the training data. In this work, we study property inference in scenarios where the adversary can maliciously control part of the training data (poisoning data) with the goal of increasing the leakage.

Previous work on poisoning attacks focused on trying to decrease the accuracy of models either on the whole population or on specific sub-populations or instances. Here, for the first time, we study poisoning attacks where the goal of the adversary is to increase the information leakage of the model. Our findings suggest that poisoning attacks can boost the information leakage significantly and should be considered as a stronger threat model in sensitive applications where some of the data sources may be malicious.

We first describe our *property inference poisoning attack* that allows the adversary to learn the prevalence in the training data of any property it chooses: it chooses the property to attack, then submits input data according to a poisoned distribution, and finally uses *black box queries* (label-only queries) on the trained model to determine the frequency of the chosen property. We theoretically prove that our attack can always succeed as long as the learning algorithm used has good generalization properties.

We then verify effectiveness of our attack by experimentally evaluating it on two datasets: a Census dataset and the Enron email dataset. In the first case we show that classifiers that recognizes whether an individual has high income (Census data) also leak information about the race and gender ratios of the underlying dataset. In the second case, we show classifiers trained to detect spam emails (Enron data) can also reveal the fraction of emails which show negative sentiment (according to a sentiment analysis algorithm); note that the sentiment is not a feature in the training dataset, but rather some feature that the adversary chooses and can be derived from the existing features (in this case the words). Finally, we add an additional feature to each dataset that is chosen at random, independent of the other features, and show that the classifiers can also be made to leak statistics about this feature; this shows that the attack can target features completely uncorrelated with the original training task. We were able to achieve above 90% attack accuracy with $9-10\%$ poisoning in all of these experiments.

I. INTRODUCTION

Machine learning is revolutionizing nearly every discipline from healthcare to finance to manufacturing and marketing. However, one of the limiting factors in ML is availability of large quantities of quality data. This has prompted calls for collaborative learning, where many parties combine datasets to train a joint model [1, 40]. However, much of this data involves either private data about individuals or confidential enterprise information. In order to use data without revealing sensitive information about the training data, the field of privacy preserving machine learning is developing fast with significant progress on technologies like differential privacy and cryptography based learning systems.

Inference Attacks: Inference attacks consider an adversary who tries to infer sensitive information about the training set by inspecting the model that is trained on it. Inference attacks have come in two main flavors: membership inference [43] and property inference attacks [2].

In a membership inference attack, an adversary tries to infer if a special instance was present in the training set that was used to train a given model. In property inference attacks, the adversary tries to infer some aggregate information about the whole training set. While there are some promising approaches for defending against membership inference attacks (e.g. differential privacy), there is no general defense mechanism known against property inference attacks and how to defend against them is still an open question. In this work, we focus on property inference attacks in collaborative learning scenarios and show that these attacks are more devastating than previously thought.

Note that the property being inferred need not be an explicit feature in the training set, nor does it need to be obviously correlated with the training set labels. For example, we will consider a property inference attack on a text based model (in particular a spam classifier), which attempts to learn the average sentiment (positive or negative) of the documents in the training dataset.

Poisoning Attacks In poisoning attacks, some part of training data (poisoning data) is carefully chosen by an adversary who wishes to make the trained model behave in his own interest. A considerable body of works [6, 47, 42, 32, 48, 5, 4] have shown that poisoning attacks can significantly hurt accuracy of ML models.

Poisoning Attacks Increasing Information Leakage In this work we initiate the study of poisoning attacks that aim at increasing the information leakage in ML models. In particular, we ask the following question:

Can adversaries boost the performance of property inference attacks by injecting specially crafted poison data in the training set?

This is a very relevant question whenever data is gathered from multiple sources, some of which may be adversarially controlled. In particular, it is relevant in collaborative machine

learning, where one party might contribute malicious data in order to learn some property about the rest of the training set.

To follow our above example, if a group of small companies pool their data to train a spam classifier, one company might contribute poisoned data in an attempt to learn about the average sentiment in the rest of the group. This could give that company a significant edge in understanding it's competitors' positions.

We note that the above question could also be asked for membership inference attacks. While that is an interesting direction, in this paper we only focus on property inference attacks. We show that poisoning can indeed increase the property leakage significantly.

Attack Model: In this paper we consider an attacker who is allowed to first submit a set of "poisoned" data of its choice, which will be combined with the victim dataset and used to train a model. Note that our adversary is oblivious to the "clean" training set sampled from other sources[1]. The attacker can then make a series of black box queries to the trained model. Note that by black box queries we mean that the adversary only gets to learn the predicted label for each query. (It does not, for example, get the confidence values that the model produces, or any internal information from the model.) Finally, as in the property inference attacks of [16, 2], the attacker's goal is to infer whether the average of a particular property is above or below a particular threshold. This is a very natural model for poisoning in the distributed learning setting where data is gathered from many sources.

Main Contributions:

- Theoretical Results: We first describe a theoretical attack that works as long as the training algorithm outputs a Bayes-optimal classifier. The high level idea of our attack is that the adversary adds poisoning points in a way that causes the behavior of the Bayes-optimal classifier to depend on the average of the target property. In particular, we show that poisoning would change the prediction of certain instances when the average of property is below the threshold. But when the average is higher than the threshold, then the poisoning does not change the prediction of those points. See Section V for details of our analysis.

  We formalize this intuition by giving a concrete attack based on our theoretical analysis in this model and analyzing it's effectiveness. Our attack is agnostic to the architecture of the trained model since it is completely black box. Note that poisoning is a crucial aspect of our theoretical analysis and the information leakage does not necessarily exist if we do not allow the adversary to poison the data.

- Experimental Results: Real training algorithms do not always output Bayes optimal classifiers, so there is a question about whether the above results hold in practice. To explore this we implement our attack on several target properties on a Census dataset and on the Enron email

---

[1]Oblivious poisoning attacks are provably weaker than non-oblivious poisoning attacks in what they can achieve [18].

dataset; in the first setting we consider a model trained to predict whether an individual with a given feature set has high income. In the second setting, we consider a spam classifier trained on the emails. The objective of the attacker in all these experiments is to distinguish whether or not the target property appears with high frequency in the dataset. Our attack can successfully distinguish various ranges of *higher* vs. *lower* frequencies, e.g. (5% from 10%), (40% from 60%), (30% from 70%). We experiment with three types of target properties:

1) Properties that are present in the training dataset as a feature, e.g., Gender and Race in the Census dataset.
2) Properties that are not present as a feature in the training data, but which may be derived from those features e.g., Negative sentiment in emails (as determined by sentiment analysis) in the Enron dataset.
3) Properties that are uncorrelated with the rest of the training data or classification task: for this experiment, we added an independently chosen random binary feature to each data entry in both the Census and the Enron data.

Experimenting with these variety of target properties demonstrate the power of our attack. In our attacks, we were able to achieve above 90% attack accuracy with about 10% poisoning in all of these experiments. We run our experiments for logistic regression and also for fully connected neural networks (See Section VII.). The whitebox attack in [17] used Gender and Race as their target properties on Census dataset, so these experiments provide a reference for comparison, and show that with moderate poisoning 5 − 10% our attack significantly improves the performance both in accuracy and running time, and of course in the access model which is fully black box. Moreover, we implemented the attack of [17] on our random features (the third type listed above), and show that the attack accuracy is barely above 50%, while our attack with 5 − 10% poisoning is able to achieve near perfect accuracy.

A. Organization

The rest of the paper is organized as follows. In Section II, we discuss the related work. In Section IV we describe our threat model and formalize it. In Section V we describe the theoretical attack for Bayes Optimal Classifiers, and in Section VI we show our concrete attack. We discuss the experiments in Section VII and conclude with some future directions to explore in Section VIII.

II. RELATED WORK

It is quite well known by now that understanding what ML models actually memorize from their training data is not trivial. As discussed above, there is a rich line of work that tries to investigate privacy leakage from ML models under different threat models. Here we provide some more detail on several of the works which seem most related to ours. For a

comprehensive survey on the other privacy attacks on neural networks, please see [21].

The global property inference attacks of [3, 17] are the most relevant to us: here the adversary's goal is to infer sensitive global properties of the training dataset from the trained model that the model producer did not intend to share. We have already described some examples above. Property inference attacks were first formulated and studied in [3]. However, this initial approach did not scale well to deep neural networks, so [17] proposed a modified attack that is more efficient. The main differences from our attack are in the threat model: 1) our adversary can poison a portion of the training data and 2) in [3, 17] the adversary has whitebox access to the model meaning that it is given all of the weights in the neural net, while our adversary has only blackbox access to the trained model as described above. We experimentally compare our attack performance and accuracy with that of [17] in Section VII.

Another closely related attack is the more recent subpopulation attack [23]. Here the adversary's goal is to poison part of the training data in such a way that only the predictions on inputs coming from a certain subpopulation in the data are impacted. To achieve this, the authors poison the data based on a filter function that specifies the target subpopulation. However, the goal of these subpopulation attacks is to attack the accuracy of the model.

In [36] the authors studied property leakage in the federated learning framework. In federated learning the process proceeds through multiple rounds. In each round each of $n > 2$ parties takes the intermediate model and uses their own data to locally compute an update. These updates are all collected by a central party and used to form the next intermediate model. The threat model in [36] is the following: n parties participate in a ML training using federated learning where one of the participant is the adversary. The adversary uses the model updates revealed in each round of the federated training and tries to infer properties of the training data that are true of a subpopulation but not of the population as a whole. We note that in this threat model, the adversary gets to see more information than on our model, so this result is not directly comparable to ours.

We also note that our work is similar to [41] in spirit, where the authors develop Bayes optimal strategies for a *grey box* membership inference attack[2]. The attack the authors study is membership-inference (as opposed to property inference), so the goal and technique of the attack and the assumptions they make about the model are completely different from ours.

Another related work is the recent work of [24] that studies poisoning attacks as a tool to evaluate differential privacy and obtain lower bounds for differential privacy of existing algorithms. This poisoning strategy improved the previous lower bound on the state of the art differentially private algorithms [25].

<hr>

[2]We note that, even though the authors claimed that there results hold for black-box attack, in reality, they addressed grey-box attacks as the adversary in their model gets the confidence scores for different classes and not just the prediction.

We also have to mention some of different type of attacks that utilize the information leakage of the learning algorithms. Membership inference attacks try to infer whether a specific example has been in the training set or not are distinct from us in that the goal of attacker is different but are similar to our setting in assumptions behind the attack [43, 9, 38, 44]. Model inversion attacks try to reconstruct part of the training set using access to the model which is a very strong attack model but only works on models that memorize too much about data [15]. Backdoor attack that try to inject backdoors through poisoning the training set and trigger the backdoors could in the test time. These attacks are similar to us as both attacks leverage poisoning [8, 19]. Another important attack models against machine learning is the study of adversarial examples where an attacker tries to add small perturbations during inference and cause the model to misclassify/mispredict the input[20, 29, 22]. Interestingly, the implications of these attacks and defenses proposed against them on privacy and membership inference is studied in [46].

Finally, we want to mention the line of work on poisoning attacks with provable guarantees. There is a line of work on poisoning attacks against *any* classifier where the goal of adversary is to increase the probability of an undesirable property over the final model [33, 30, 34, 12, 35, 31]. Since these attacks are very general, the adversary can set this undesirable property to be about the leakage of final model. However, these attacks require the adversary to know the "clean" dataset before injecting/replacing the poisoning examples. Moreover, they require the learning algorithm to be deterministic and also have some original probability of the bad property happening even when data is provided in a benign fashion. Our attack is distinct from those attacks as it is oblivious to the clean dataset. Also, our attack only requires the learning algorithm to have some generalization properties and does not need it to be deterministic or having some original chance of outputting a leaky model. Another line of work tries to show poisoning attacks with provable guarantees against learning algorithms that use convex loss minimization [47, 27, 48]. Our attack is distinct from these results in that our attacks have provable guarantees against *any* learning algorithm that has good generalization and also it does not need the knowledge of clean training set. On the positive side, the work of [10] showed the power of robust statistics in overcoming poisoned data in polynomial-time. These works led to an active line of work [11, 39, 28] exploring the possibility of robust statistics over poisoned data with algorithmic guarantees. These results do not apply to our settings as they either are about robust parameter estimation of distributions or about accurate classification in presence of outliers.

III. PROPERTY INFERENCE ATTACKS

There has been a series of work looking at to what extent a model leaks information about a certain *individual record* in the training set, including work on using differential privacy [13] to define what it means for a training algorithm to preserve privacy of these individuals and technically how that can be achieved.

However, leaking information on individuals is not the only concern in this context. In many cases even the aggregate information is sensitive.

This type of aggregate leakage inspires a line of work started in [3, 17] that looks at *property inference* attacks, in which the attacker is trying to learn aggregate information about a dataset. In particular, we focus here, as did [17], on an attacker who is trying to determine the frequency of a particular property in the dataset used to train the model. Notice that this type of aggregate leakage is a global property of the training dataset and is not mitigated by adding differential-privacy mechanisms.

Does property inference pose an important threat model? Property inference attacks could reveal very sensitive in formation about the dataset. To illustrate the importance of the attack model, we provide some examples of such sensitive information that could be revealed. For further discussion on the importance of these attacks we refer the reader to previous work of [16] and [2].

Example 1. *Imagine a company wants to use its internal emails to train a spam classifier. Such a model would be expected to reveal which combination of words commonly indicate spam, and companies might be comfortable sharing this information. However, using a property inference attack, the resulting model could also leak information about the aggregate sentiment of emails in the company, that could potentially be very sensitive. For example, if the sentiment of emails in the company turn negative near the financial quarter, it could mean that the company is performing below expectations.*

Example 2. *Similarly, a financial company might be willing to share a model to detect fraud, but might not be willing to reveal the volume of various types of transactions.*

Example 3. *Or a number of smaller companies might be willing to share a model to help target customers for price reductions etc, however such companies might not be willing to share specific sales numbers for different types of products.*

Property leakage from poisoned datasets One of the questions that is not addressed in previous work on property inference attacks is scenarios where adversary can contribute to the part of the training set. This could occur either because one of the parties in collaborative training behaves adversarially, or because an adversary can influence some of the input sources from which training data is collected (e.g. by injecting malware on some data collection devices). Specifically, the adversary can try to craft special poisoning data so that it can infer the specific property that it has in mind. Note that this is not prevented by any of the cryptographic or hardware assisted solutions: in all of these there is no practical way to guarantee that the data that is entered is actually correct.

This type of *poisoning* attack has been extensively studied in the context of security of ML models, i.e., where the goal of the attacker is to train the model to miss-classify certain datapoints [6, 42, 32], but to the best of our knowledge ours

is the first work that looks at poisoning attacks that aim to compromise privacy of the training data.

Black box or white box model access The information leakage of machine learning models could be studied in both white-box and black-box setting. In this paper, we consider the *black box* model, where the attacker is only allowed to make a limited number of queries to the trained model. We show that these attacks can be very successful. "Black box" attacks is sometimes used to refer to attacks which also have access to model's confidence values on each query [41]. We emphasize here that we use the stricter notion of black box and our attacker will use only the model predictions. This type of attack is studied independently in [9] where they study "label-only" membership inference attacks.

IV. THREAT MODEL

Before going through the threat model, we introduce some useful notation.

Notation. We use calligraphic letter (e.g $\mathcal{T}$) to denote sets and capital letters (e.g. D) to denote distributions. We use (X, Y) to denote the joint distribution of two random variables (e.g. the distribution of labeled instances). To indicate the equivalence of two distributions we use $D_1 \equiv D_2$. By $x \leftarrow X$ we denote sampling x from X and by $\Pr_{x \leftarrow X}$ we denote the probability over sampling x from X. We use $\text{Supp}(X)$ to denote the support set of distribution X. We use $p \cdot D_1 + (1 - p) \cdot D_2$ to denote the weighted mixture of D_1 and D_2 with weights p and $(1 - p)$.

Property Inference: To analyze property inference, we follow the model introduced in [16]. Consider a learning algorithm $L \colon (\mathcal{X} \times \mathcal{Y})^* \to \mathcal{H}$ that maps datasets in $\mathcal{T} \in (\mathcal{X} \times \mathcal{Y})^*$ to a hypothesis class $\mathcal{H}$. Also consider a Boolean property $f \colon \mathcal{X} \to \{0, 1\}$. We consider adversaries who aim at finding information about the statistics of the property f over dataset $\mathcal{T} \in (\mathcal{X} \times \mathcal{Y})^*$, that is used to train a hypothesis $h \in \mathcal{H}$. In particular, the goal of the adversary is to learn information about $\hat{f}(\mathcal{T})$ which is the fraction of data entries in $\mathcal{T}$ that has the property f over data entries, namely $\hat{f} = \mathbf{E}_{(x,y) \leftarrow \mathcal{T}}[f(x)]$. More specifically the adversary tries to distinguish between $\hat{f}(\mathcal{T}) = t_0$ or $\hat{f}(\mathcal{T}) = t_1$ for some $t_0 < t_1 \in [0, 1]$. We are interested in the black-box setting where the adversary can only query the trained model on several points to see the output label. I.e. the adversary does not get the confidence values for his queries, or any information about the parameters of the model. This model is also known as label-only attacks and has been recently explored in the context of membership-inference attacks [9].

Formal Model: To formalize this, we use distributions D_-, D_+ to denote the underlying distribution of the dataset for instances with $f(x) = 0$ and $f(x) = 1$ respectively. Then we consider two distributions made by mixing D_-, D_+ at different ratios, i.e.,

$$D_t \equiv t \cdot D_+ + (1 - t) \cdot D_-$$

D_t is the distribution where t fraction of the points have $f(x) = 1$. The adversary's goal is to distinguish between D_{t_0}

and D_{t_1}, for some $t_0 < t_1$, by querying (in a black box way) a model M that is trained on one of these distributions. In this attack, as in previous work [3, 17] we assume that the adversary can sample from D_-, D_+.

Property Inference with Poisoning: We consider the poisoning model where adversary can contribute pn "poisoned" points to a n-entry dataset T that is used to train the model. To the best of our knowledge, this is the first time that poisoning attacks against privacy of machine learning are modeled and studied.

In order to measure the power of adversary in this model we define the following adversarial game between a challenger C and an adversary A. Our game mimics the classic indistinguishability game style used in cryptographic literature. As described above, L is the learning algorithm, n is the size of the training dataset of which p fraction are poisoned points selected by an adversary. D_-, D_+ are the distributions of elements x with $f(x) = 0, 1$ respectively, and the goal of the attacker is to tell whether the fraction of points in the victim dataset that are from D_+ is t_0 or t_1.

Assumptions on the Adversary's Knowledge The adversary has access to the following:

Sample access to conditional distributions

D_+, D_- and X_+ (formally defined in Def. 8). Note that the adversary has no knowledge about the distribution of the target feature. It is also worth noting that this assumption is the same as what is used in previous property and membership inference attacks [43, 17, 9, 38, 37, 45]. In many cases this is a very reasonable assumption. For instance, in the Enron email example, it is not hard for an adversary to gather sample emails with positive and negative sentiments. In other scenarios it might be the case that some subset of the data has been made public, or that the adversary has some data that follows a similar distribution.

Blackbox access to the trained model

The adversary can query the trained model to get output labels alone and nothing else.

Training algorithm and the features

This is completely reasonable since we are in the collaborative setting, where the adversary is one of the collaborators and each collaborator has the right to know how their data will be processed. It is also worth noting that most practical techniques for performing collaborative training in a decentralized and privacy-preserving way (e.g., secure multi-party computation, SGX) will require each collaborator to know the training algorithm and the features. Finally, it is never a good idea to try to achieve privacy by hiding the algorithm details as it is rather easy to obtain these information, e.g. see [49].

PIWP$(L, n, p, D_-, D_+, t_1, t_0)$:

---

1) C select a bit $b \in \{0, 1\}$ uniformly at random. then samples a dataset of size $(1-p) \cdot n$: $\mathcal{T}_{\text{clean}} \leftarrow D_{t_b}^{(1-p)n}$
2) Given all the parameters in the game, A selects a poisoning dataset $\mathcal{T}_{\text{poison}}$ of size pn and sends it to C.
3) C then trains a model $M \leftarrow L(\mathcal{T}_{\text{poison}} \cup \mathcal{T}_{\text{clean}})$.
4) A adaptively queries the model on a sequence of points $x_1, \ldots, x_m$ and receives $y_1 = M(x_1), \ldots, y_m = M(x_m)$.
5) A then outputs a bit b' and wins the game if $b = b'$.

We aim to construct an adversary that succeed with probability significantly above 1/2.

V. ATTACK AGAINST BAYES-OPTIMAL CLASSIFIERS

In this section, we will introduce a theoretical attack with provable guarantees for Bayes-optimal classifiers. A Bayes-optimal classifier for a distribution $D \equiv (X, Y)$ is defined to be a classifier that provides the best possible accuracy, given the uncertainty of D. Below, we first define the notion of Risk for a predictor and then define Bayes error and Bayes-optimal classifier based on that.

Definition 4 (Risk of predictors)**.** *For a distribution $D \equiv (X, Y)$ over $\mathcal{X} \times \{0, 1\}$ and a predictor $h: \mathcal{X} \to \{0, 1\}$, the risk of h is defined as $\mathsf{Risk}(h, D) = \Pr_{(x,y) \leftarrow D}[h(x) \neq y]$.*

Definition 5 (Bayes Error and Bayes-optimal classifier)**.** *Let $D \equiv (X, Y)$ be a distribution over $\mathcal{X} \times \{0, 1\}$. The Bayes error of D is defined to be the optimal risk for any deterministic predictor of $\{0, 1\}$ from $\mathcal{X}$. Namely,*

$$\mathsf{Bayes}(D) = \inf_{h: \ \mathcal{X} \to \{0,1\}} \mathsf{Risk}(h, D).$$

Also the Bayes-optimal classifier for D is defined to be a hypothesis h_D^* that minimizes the error over the distribution. Such classifier always exists if the support of Y is a finite set (which is the case here since $\mathrm{Supp}(Y) = \{0, 1\}$.) In particular, the following function is a Bayes-optimal classifier for any such D

$$\forall x \in \mathcal{X} : h_D^*(x) = \operatorname*{argmax}_{y \in \{0,1\}} \Pr[Y = y \mid X = x].$$

The Bayes error is the best error that a classifier can hope for. High performance learning algorithms try to achieve error rates close to Bayes error by mimicking the behavior of Bayes-optimal classifier. Below, we assume a learning algorithm that can learn the almost Bayes-optimal classifier for a class of distributions, and will show that even such a high quality learning algorithm is susceptible to attack. Let us first define the notion of approximate Bayes optimal learning algorithms.

Definition 6 ((ε, δ)-Bayes optimal learning algorithm)**.** *For two functions $\varepsilon : \mathbb{N} \to [0, 1]$ and $\delta : \mathbb{N} \to [0, 1]$, a learning algorithm L is called a (ε, δ)-Bayes optimal classifier for a*

distribution $D \equiv (X, Y)$ iff for all $n \in \mathbb{N}$, given a dataset $\mathcal{T} \leftarrow D^n$ with n samples from D, the learning algorithm outputs a model h such that

$$\mathsf{Risk}(h, D) \leq \mathsf{Bayes}(D) + \varepsilon(n).$$

with probability at least $1 - \delta(n)$ over the randomness of samples from the distribution.

Note that ϵ and δ are usually decreasing functions of n that can converge to 0. Now, we are ready to state our main theorem. But before that, we need to define the notion of certainty of a point which defines the (un)certainty of the response variable on a particular point x. We use this notion to identify the points that have a lot of ambiguity. We will see in our theorem below that if we have enough ambiguous points, we can run our attack.

Definition 7 (Signed certainty). *For a distribution $D \equiv (X, Y)$ over $\mathcal{X} \times \{0, 1\}$, the (signed) certainty of a point $x \in \mathcal{X}$ according to D is defined by*

$$\mathsf{crt}(x, D) = 1 - 2 \Pr[Y = 1 \mid X = x].$$

Definition 8 (Class of distributions induced by a property). *Let $f : \mathrm{Supp}(X) \to \{0, 1\}$ be a property and $D \equiv (X, Y)$ be a distribution of labeled instances. Let $X_+ \equiv X \mid f(X) = 1$ and $D_+ \equiv (X, Y) \mid f(X) = 1$ and $D_- \equiv (X, Y) \mid f(X) = 0$. We use $\mathcal{D}_f$ to denote the following class of distributions:*

$$\mathcal{D}_f = \{\alpha_1 \cdot (X_+, 1) + \alpha_2 \cdot D_+ + \alpha_3 \cdot D_-$$
$$\text{where } \alpha_i \in [0, 1], \sum_{i=1}^{3} \alpha_i = 1\}.$$

We are now ready to present our main theorem which describes conditions where an almost Bayes optimal algorithm is vulnerable to attack. We will prove this theorem in the following section.

Theorem 9. *Let $D \equiv (X, Y)$ be a distribution over $\mathcal{X} \times \{0, 1\}$ and $f : \mathcal{X} \to \{0, 1\}$ be a property over its instances. Let $D_+ \equiv (X_+, Y_+) \equiv (X, Y) \mid f(X) = 1$ and $D_- \equiv (X_-, Y_-) \equiv (X, Y) \mid f(X) = 0$ be conditional distributions based on property f. Consider a learning algorithm L that is (ε, δ)-Bayes optimal for class $\mathcal{D}_f$. For any $p, t_0 < t_1 \in [0, 1]$, if there exist $\tau \in [0, 1]$*

$$\Pr_{x \leftarrow X} \left[\frac{p + 2\tau \cdot t_1}{t_1(1 - p)} < \mathsf{crt}(x, D) \leq \frac{p - 2\tau \cdot t_0}{t_0(1 - p)} \right.$$
$$\left. \wedge \quad f(x) = 1 \right] > \frac{2\epsilon(n)}{\tau}$$

then there is an adversary A who wins the security game $\mathsf{PIWP}(n, L^*, D_-, D_+, p, t_0, t_1)$ *with probability at least* $1 - 2\delta(n)$.

Theorem 9 states that our attack will be successful in distinguishing D_{t_0} from D_{t_1} if there are enough points in the distribution with high uncertainty and the learning algorithm is Bayes-optimal for a large enough class of distributions.

Remark 10. *One can instantiate Theorem 9 by replacing f with $1 - f$. In this case, instead of t_0 and t_1 we need to work with $1 - t_0$ and $1 - t_1$. On the other hand, we can also flip the labels and use the distribution $(X, 1 - Y)$ instead of (X, Y). Using these replacements, we can get four different variants of the theorem with different conditions for the success of the attack. In Section VI we will see that in different scenarios we use different variants as that makes the attack more successful.*

Remark 11 (On the possibility of defending our attack with differential privacy). *Theorem 9 states that our attack will be successfully against* any *learning algorithm that provides strong generalization bounds for both benign and poisoned distributions. Differential privacy will not treat different distributions differently and we believe the only way it could be effective is by dropping the accuracy for both benign and poisoned distribution. Also, we have to note that differential privacy does not provide any guarantees in our setting as the information leaked through our attack is a global property of the dataset and not related to an individual example in the training set.*

A. Attack Description

In this section we prove Theorem 9. We first describe an attack and then show how it proves Theorem 9.

The rough intuition behind the attack is the following. If an adversary can produce some poisoning data at the training phase to introduce correlation of the target property with the label, then this will change the distribution of training examples. This will change the resulting classifier as well because the learning algorithm is almost Bayes optimal and should adapt to distribution changes. This change will cause the prediction of the uncertain cases (i.e., those which occur in the training set almost equally often with label 0 and 1) to change. The adversary can choose the poisoning points in a way that the change of prediction is noticeably different for certain points between the cases when the target property occurs with frequency t_0 vs t_1. In the rest of section, we will show how adversary selects the poisoning points. Then we will see that the behavior of resulting model should be different on points that satisfy the conditions in Theorem 9, depending on the distribution used during the training. At the end, we argue that by querying these points, the adversary can distinguish between the case of t_0 and t_1.

Let the original data distribution of clean samples be $D \equiv (X, Y)$. Our adversary A will pick the poison data by i.i.d. sampling from a distribution $D_A \equiv (X_A, Y_A)$. Note that this adversary is weak in a sense that it does not control the poison set but only controls the distribution from which the poison set is sampled. The resulting training dataset will be equivalent to one sampled from a distribution $\tilde{D}$ such that $\tilde{D}$ is a weighted mixture of D and D_A. More precisely,

$$\tilde{D} \equiv \begin{cases} D, & \text{With probability } (1 - p) & \text{[Case I: No poisoning]}, \\ D_A & \text{With probability } p & \text{[Case II: Poisoning]} \end{cases}$$

Now we describe the distribution D_A. To sample poisoning points, adversary first samples a point from X conditioned on having the property f. For the label, the adversary always chooses label 1. So we have $D_A \equiv (X_+, 1)$. We will see how this simple poisoning strategy can help the adversary to infer the property.

B. Evaluating the attack

In this section, we evaluate the effect of the poisoning strategy above on the Classifiers. We describe the evaluation steps here and defer proofs to the appendix.

Effect of poisoning on the distribution

Let $(\tilde{X}, \tilde{Y})$ be the joint distribution of samples from $\tilde{D}$. First we calculate $\Pr[\tilde{Y} = 1 \mid \tilde{X} = x]$ to see the effect of poisoning on the distribution. Let E be the event that the point is selected by adversary, namely, the second case in the description of $\tilde{D}$ happens. Let $t = \Pr[f(X) = 1]$, we prove the following claim.

Claim 12. *For any $x \in \mathcal{X}$ such that $f(x) = 1$ we have*

$$
\Pr[\tilde{Y} = 1 \mid \tilde{X} = x] = \frac{p}{p + t(1-p)}
$$
$$
+ \frac{t(1-p)}{p + t(1-p)} \cdot \Pr[Y = 1 \mid X = x].
$$

As a corollary to this claim, we prove that

Corollary 13. *For any x such that $f(x) = 1$ and for any $\tau \in \mathbb{R}$ we have $\Pr[\tilde{Y} = 1 \mid \tilde{X} = x] \geq \frac{1}{2} + \frac{\tau \cdot t}{p + t(1-p)}$ if and only if $\mathsf{crt}(x) \leq \frac{p - 2\tau \cdot t}{t(1-p)}$*

Claim 12 and Corollary 13 show how the adversary can change the distribution. We now want to see the effect of this change on the behavior of the Bayes optimal classifier.

Effect on the Bayes Optimal Classifier

Consider the algorithm L that is (ε, δ)-Bayes optimal on all linear combinations of distributions D_+, D_- and D_A (as stated in Theorem 9). Therefore, on a dataset $\tilde{\mathcal{T}} \leftarrow \tilde{D}^n$, the algorithm L will output a model $\tilde{h} = L(\tilde{T})$ that with probability at least $1 - \delta(n)$ has error at most $\mathsf{Bayes}(\tilde{D}) + \varepsilon(n)$. Consider joint distribution $(\tilde{X}, \tilde{Y})$ such that $\tilde{D} \equiv (\tilde{X}, \tilde{Y})$. The following claim shows how the adversary can exploit the dependence of the probabilities on t and infer between t_0 and t_1 by using special points that have high uncertainty.

Claim 14. *Let L be a (ϵ, δ)-Bayes optimal learning algorithm for $\tilde{D} \equiv (\tilde{X}, \tilde{Y})$. Consider an event C_τ defined on all $x \in \mathcal{X}$ such that $C_\tau(x) = 1$ iff $f(x) = 1$ and*

$$
\frac{p + 2\tau \cdot t_1}{t_1(1-p)} < \mathsf{crt}(x) \leq \frac{p - 2\tau \cdot t_0}{t_0(1-p)}.
$$

If there exist a $\tau \in [0, 1]$ and $\gamma \in [0, \frac{1}{2}]$ such that we have $\Pr[C_\tau(X) = 1] \geq \frac{\varepsilon(n)}{\tau \cdot (1 - 2\gamma)}$ then if $t = \Pr[f(X) = 1] = t_0$ we have

$$
\Pr_{\substack{S \leftarrow \tilde{D}^n \\ \tilde{h} \leftarrow L(S)}} \left[\Pr_{x \leftarrow X | C_\tau(x) = 1} \left[\tilde{h}(x) = 1 \right] \geq 0.5 + \gamma \right] \geq 1 - \delta(n)
$$

and if $t = \Pr[f(X) = 1] = t_1$

$$
\Pr_{\substack{S \leftarrow \tilde{D}^n \\ \tilde{h} \leftarrow L(S)}} \left[\Pr_{x \leftarrow X | C_\tau(x) = 1} \left[\tilde{h}(x) = 1 \right] \leq 0.5 - \gamma \right] \geq 1 - \delta(n).
$$

Putting it together

Using Claim 14, we will finish the proof of Theorem 9. Intuitively, the adversary wants to calculate the probability $\Pr_{x \leftarrow X | C_\tau(x) = 1} \left[\tilde{h}(x) = 1 \right]$. For doing this we use standard sampling methods to estimate the probability.

- The adversary first samples $m = \frac{-\log(\delta(n))}{2\gamma^2}$ samples from $(X, Y) \mid C_\tau(X) = 1$. In order to do this, adversary needs to sample roughly $m \cdot \frac{\tau(1 - 2\gamma)}{\varepsilon(n)}$ points from X_+.
- the adversary then calls $\tilde{h}$ on all the points and calculates the average of the prediction as ρ.

In the case where $\mathsf{Risk}(h, \tilde{D}) \leq \epsilon(u)$ and $t = t_0$, Using claim 14 and Chernoff-Hoefding inequality we have

$$
\Pr[\rho < 0.5] \leq \delta(n).
$$

Similarly in the case where $t = t_1$, and $\mathsf{Risk}(h, \tilde{D}) \leq \epsilon(n)$, we have

$$
\Pr[\rho > 0.5] \leq \delta(n).
$$

Therefore, the adversary only checks if $\rho > 0.5$ and based on that decides if $t = t_0$ or $t = t_1$. The probability of this test failing is at most $2\delta(n)$ as there is at most $\delta(n)$ probability of the learning algorithm failing in producing a good classifier and $\delta(n)$ probability of failure because of the Chernoff-Hoefding. Therefore, with probability at least $(1 - 2\delta(n))$ the adversary can infer whether $t = t_0$ or $t = t_1$.

VI. A CONCRETE ATTACK

Here we describe the concrete attack we use in our experiments. We note that there are many possible variations on this attack; what we have presented here is just one configuration that demonstrates that the attack is feasible with high accuracy. We leave exploration of some of the other variations to future work.

Selecting Poisoning Points Recall that the attacker is assumed to be able to sample from D_- and D_+, the distribution of items with $f(x) = 0$ and with $f(x) = 1$. As in the theoretical attack described in Section V, the poisoned data is generated by sampling from $D_+ \equiv (X_+, Y_+)$ and introducing correlation between the target feature $f(x)$ and the label by injecting poisoning points of form $(X_+, 1)$.

Note that the attack described in the Section VI could be achieved in 4 different forms. Namely, the adversary can use any of the possible combinations $(X_-, 1)$, $(X_-, 0)$, $(X_+, 0)$ and $(X_-, 1)$ for poison data. In algorithm 3 we show how we choose between these strategies. In particular, when values of t_0 and t_1 are large, we try to attack $1 - f$ instead of f. The logic behind this choice is that it is easier to impose a correlation

between the property and the label, when the property is rare in the distribution. As an example, consider a spam detection scenario where the adversary wants to impose the following rule on the spam detection: all Emails that contain the word "security" must be considered spam. It would be much easier for the adversary to impose this rule when there are very only a few Emails containing this word in the clean dataset. In other words, if this rule is added to the spam detector, the accuracy of the spam detector on the rest of the emails would not change significantly as there are not many email containing the word "security". On the other hand, we select the correlation in the direction that is not dominant in the data. Namely, if we have $Pr_{(x,y)\leftarrow(X,Y)}[y = 1|f(x) = 1] \geq 0.5$ then we either use $(X_+, 0)$ or $(X_-, 1)$ based on the values of t_0 and t_1. The intuition behind this choice is that we always select the correlation that happens less often in the training set so that the poisoning distribution is more distinct from the actual distribution and hence makes a larger change in the distribution. Note that we can just stick to one of these four options and still get successful attacks for most scenarios but our experiments shows that this is the most effective way of poisoning. Also, it is important to note that Theorem 9 could be extended to all of these attacks with slightly different conditions. The details of the poisoning algorithm are described in Algorithm 1.

Selecting Query points: The next challenge here is in choosing the query points. As in Section V, we want to find query points whose certainty falls in a range close to 0. For instance, if the poisoning rate $p = 0.1$ and we want to distinguish between $t_0 = 0.3$ and $t_1 = 0.7$, the Theorem suggests that we should be querying the points whose certainty falls between $[\approx 0.15, \approx 0.37]$. In order to do this, we need to first calculate the certainty. Since we only have sampling access to the distribution, we do not know the exact certainty. Instead we approximate the certainty by estimating the probability that the label is 0 or 1 through training an ensemble of models and then evaluating all of them on a point. In particular, for a point x we estimate $\Pr[Y = 1 \mid X = x]$ using the fraction of models in the ensemble that predict 1. Then we use this estimate to calculate the certainty. score

The way we estimate the certainty is obviously prone to error. To cope with the error, we actually work with a larger range than what is suggested by our theoretical result. In out attack, we fix the range of certainty to $[-0.4, 0.4]$ and query the points whose certainty falls in this interval. Although this range might be larger (and sometimes smaller) than what our Theorem suggests, but we still get good results in our attack. The reason behind this is that in the next step of the attack, which is the inference phase, we filter the query points and only look at the important ones. So if there are query points that are not relevant, they will be ignored in the inference.

Guessing the Fraction: At the end, the adversary must use the results of its queries to come up with the prediction of whether $t = t_0$ or $t = t_1$. The theoretical attack suggests that we should just take the average of all responses and

predict based on whether or not the average is less than 0.5. However, as we pointed out before, we cannot use the exact range suggested by the theorem because we do not have access to exact certainty. To get around this issue, we train a linear model for the attack to identify the queries that are actually relevant in predicting whether $t = t_0$ or $t = t_1$. Here we use a shadow model approach: we sample multiple datasets $\mathcal{T}_1^0, \ldots, \mathcal{T}_k^0$ and $\mathcal{T}_1^1, \ldots, \mathcal{T}_k^1$ where each for each $i \in [k]$ we have $\Pr_{(x,y)\leftarrow\mathcal{T}_k^0}[f(x) = 1] = t_0$ and $\Pr_{(x,y)\leftarrow\mathcal{T}_k^1}[f(x) = 1] = t_1$. Now we use the poisoning dataset $\mathcal{T}_{poison}$ and the query dataset $\mathcal{T}_q$ from the previous steps of the attacks as follows: We first use the poisoning set to poison all the datasets and train multiple models $(M_1^0, \ldots, M_k^0)$ and $(M_1^1, \ldots, M_k^1)$ where $M_0^i = L(\mathcal{T}_0^i \cup \mathcal{T}_p)$ and $M_0^i = L(\mathcal{T}_0^i \cup \mathcal{T}_p)$. After that we query each of the models on each of the chosen query points. Finally, we generate a training set to train the attacker's model where the query responses are the features and the label are 0 or 1 depending on whether the "shadow" model used was from $\mathcal{T}_i^0$ or $\mathcal{T}_i^1$, and train a linear model M_A on this set. The adversary then uses the query points to query the real target model, feeds the responses to M_A, and outputs whatever M_A produces as it's final guess for whether $t = t_0$ or $t = t_1$.

Here we describe the concrete attack for distinguishing $D_{.3}$ from $D_{.7}$ with initial victim training set size n and poisoning rate p (i.e. pn poisoned points out of n total training points). Recall that the attacker has sampling access to D_-, D_+, the distribution of samples with $f(x) = 0$ and $f(x) = 1$ respectively. The attack will proceed as follows:

Algorithm 1 Choosing poisoning data

Input
    f The property being attacked
    t_0, t_1 Possible fractions of instances with property f
    p Poisoning ratio
    n Size of training set
    D_+, D_- Sampling oracle to distribution of instances with and without property f

Output
    $\mathcal{T}_{poison}$ Set of poisoning examples

1: **if** $(t_0 + t_1) < 1$ **then**
2:      Sample $m = p \cdot n$ examples from D_+,
3:
$$T = \{(x_1, y_1), \ldots, (x_m, y_m)\} \leftarrow D_+^m$$
4: **else**
5:      Sample m examples from D_-,
6:
$$T = \{(x_1, y_1), \ldots, (x_m, y_m)\} \leftarrow D_-^m$$
7: $\alpha = \frac{\sum_{i=1}^m y_i}{m}$.
8: **if** $\alpha > 0.5$ **then**
9:      $\mathcal{T}_{poison} = \{(x_1, 0), \ldots, (x_m, 0)\}$
10: **else**
11:      $\mathcal{T}_{poison} = \{(x_1, 1), \ldots, (x_m, 1)\}$
12: Output $\mathcal{T}_{poison}$

VII. Experimental Evaluation

Here we evaluate the performance and the accuracy of our attack described in Section VI.

Algorithm 2 Choosing the black box queries
___

Input

    r   number of models in ensemble

    q   Number of black-box queries

    D_+, D_- Sampling oracle to distribution of instances with and without property f

Output

    $\mathcal{T}_q$ Set of black-box queries

1: Sample a thousand data sets $\mathcal{T}_1, \ldots, \mathcal{T}_r$ each composed half of elements sampled from D_- and half of elements sampled from D_+.

2: Train $M_1, \ldots, M_r$ using $\mathcal{T}_1, \ldots, \mathcal{T}_r$.

3: Set $\mathcal{T}_q = \emptyset$.

4: **while** $|\mathcal{T}_q| < q$ **do**

5:     sample $x \leftarrow \frac{1}{2}D_- + \frac{1}{2}D_+$

6:     if

$$|1 - 2\frac{\sum_{i=1}^{r} M_i(x)}{r}| \leq 0.4$$

    $\mathcal{T}_q = \mathcal{T}_q \cup \{x\}$.

7: Set $\mathcal{T}_q = \mathcal{T}_q \cup \mathcal{T}_{\text{poison}}$

8: Output $\mathcal{T}_q$ as the set of black box queries to make.
___

Algorithm 3 Guessing the fraction of samples with property f
___

Input

    f The property being attacked

    t_0, t_1 Possible fractions of instances with property f

    p Poisoning ratio

    n Size of training set

    k Number of shadow models

    $\mathcal{T}_{\text{poison}}$ The set of poisoning data

    $\mathcal{T}_q$ The set of black-box queries

    D_+, D_- Sampling oracle to distribution of instances with and without property f

Output

    b' A bit that predicts whether $t = t_0$ or $t = t_1$.

1: Sample data sets $\mathcal{T}_1^0, \ldots, \mathcal{T}_k^0$ with size n from D_{t_0} and $\mathcal{T}_1^1, \ldots, \mathcal{T}_k^1$ with size n from D_{t_1}. (Note that the adversary can generate samples from these distributions given sampling access to D_-, D_+: e.g. to sample from D_{t_0}, first choose bit b from a distribution that is 1 with probability t_0, then sample from D_b.)

2: Train $M_1^0, \ldots, M_k^0$ using $\mathcal{T}_1^0 \cup T_{\text{poison}}, \ldots, \mathcal{T}_k^0 \cup T_{\text{poison}}$ and $M_1^1, \ldots, M_k^1$ using $\mathcal{T}_1^1 \cup T_{\text{poison}}, \ldots, \mathcal{T}_k^1 \cup \mathcal{T}_{\text{poison}}$.

3: Query all models on T_q to get $R_1^1, \ldots, R_k^1$ and $R_1^0, \ldots, R_k^0$.

4: Construct a dataset

$$\{(R_1^1, 1), \ldots, (R_k^1, 1), (R_1^0, 0), \ldots, (R_k^0, 0)\}$$

and train a linear model with appropriate regularization on it to get M_A (We use ℓ_2 regulizer with weight $2 \cdot \sqrt{1/k}$)

5: Query the target model on $\mathcal{T}_q$ to get R_q, evaluate $M_A(R_q)$, and output the result.
___

A. Experimental Setup

DataSets We have run our experiments on two datasets:

- **Census:** The primary dataset that we use for our experiments is the US Census Income Dataset [14]. The US Census Income Dataset contains census data extracted from the 1994 and 1995 population surveys conducted by the U.S. Census Bureau. This dataset includes 299,285 records with 41 demographic and employment related attributes such as race, gender, education, occupation, marital status and citizenship. The classification task is to predict whether a person earns over $50,000 a year based on the census attributes.[3]

- **Enron:** The second dataset that we use to validate our attack is the Enron email dataset [26]. This dataset contains 33717 emails. The classification task here was to classify an email as "Spam" or "Ham" based on the words used in the email.

Target Property In Table I, we summarize the features we experimented with. In all these experiments, the attacker's objective is to distinguish between two possible values for the frequency of the target feature. Below is a summary list of all these properties.

DataSet	Target Feature	Distinguish between
Census	Random binary	0.05 vs 0.15
Census	Gender	0.6 vs 0.4 female
Census	Race	0.1 vs 0.25 black
Enron	Random binary	0.7 vs 0.3
Enron	Negative sentiment	0.10 vs 0.05

TABLE I

TARGET FEATURES. FOR EACH FEATURE WE USE TWO DIFFERENT RATIOS CLOSE TO THE ACTUAL RATIOS IN THE DATASET (EXCEPT FOR RANDOM FEATURES WHICH ARE NOT PRESENT IN THE DATASET) IN OUR EXPERIMENTS.

- **Random:** To understand the power of this attack on a feature that is completely uncorrelated to the classification task (which one might naturally think should not be leaked by an ideal model), we did a set of experiments where we added a random binary feature to both Census and Enron datasets and set that as the target feature that the adversary wants to attack. Note that this feature is not correlated with any other feature in the dataset and the model should not depend on it to make its decision. This is backed up by our experiments in that, as we will see, the attack of [17], which uses no poisoning does not perform better than random guessing on this property.

- **Gender:** Gender is a boolean feature in census data which takes values "Male" and "Female". This feature was also used in the work of [16, 17], so it allows us to directly compare our work with theirs.

- **Race:** Another feature that we attack in the census dataset is Race. In our attack we tries to infer between two different ratios of "Black" race in the dataset. Again, we chose this for purposes of comparison with previous work.

- **Negative Sentiment:** In one of our target properties, we try to infer the fraction of emails in the Enron email dataset that have negative sentiment. To do this, we use the sentiment analysis tool in *python nltk* to identify emails with positive and negative sentiment. Note that unlike all the other target properties, the negative sentiment feature is not present as a feature in the dataset which makes it intuitively seem harder for the attacker to infer. However, as we will see in our experiments, the attacker can still attack this property.

___

[3]We used the census dataset as is, as compared to [17, 16] where they preprocess the census dataset and run their experiments with balanced labels (50% low income and 50% high income). We notice that in the original dataset, the labels are not balanced (around 90% low income and 10% high income).

B. Black-box queries

As mentioned before, we are interested in the information leakage caused by black-box access to the model. Namely, the adversary can query the model on a number of points and infer information about the target property using the label prediction of the model on those queries (See Section IV for more details). In a concurrent work [9], also explored this kind of black-box access in the context of membership inference attacks. Our model does not require any other information other than predicted label (e.g. no confidence score, derivative of the model, etc). The query points are selected according to algorithm 2 in such a way that they have low certainty. For Enron experiments, we use 500 query points and for census data experiments we use 1000 query points. The reason that we use different numbers for the Enron and census experiment is that Enron dataset contains fewer of "uncertain" points; we used almost all the points that fall into our range.

C. Target model architectures

Most of our experiments use logistic regression as the model architecture for training. The main reason we picked logistic regression was because it is much faster to train compare to Neural Networks. However, we also have a few experiments on the more complex models. In particular, we test our attack on fully connected neural networks with up to 6 hidden layers (See Table II). We note that since our attack is black-box, we do not need any assumption over the target model architecture other than the fact that it will have high accuracy (we still need the adversary to know the architecture for the attack to work.). This is again in contrast with the previous work of [17] that only works on fully connected neural networks. This is an important distinction as it shows that the attacker is not relying on extensive memorization due to large number of parameters.

D. Shadow model training

Our shadow model training step is quite simple. As described in Section VI, we train a series of shadow models with a fixed poisoning set. We hold out around $\frac{1}{3}$ of the dataset (For both Enron and Census datasets) training the shadow models. This held out part will not effect any of the models that we test our attacker's accuracy on. After training the shadow models, we query them and train a simple linear attack model over the predictions on the queries. We use a linear model to train the attack model since our theoretical results suggest a simple linear model (which just takes the uniform average) over the queries would be enough to make a correct prediction. We use ℓ_2 regularization for our linear models to get better generalization and also reduce the number of effective queries as much as possible. Note that this choice of simple linear models is contrast with the attack of [16] and [17]. that use complex models (e.g. set neural networks) to train the attack model; this is one of the reasons that our attack is faster.

E. Performance of our attack

In the following experiments, we evaluate the performance of our attack and compare it with the attack of [17]. In the rest of the manuscript, we denote the attack of [17] as WBAttack. . We first evaluate how the different parameters, namely, poisoning rate, training set size, number of shadow models (defined in Sec IV) and the complexity of the target model affect its accuracy. To understand the effect of each parameter, for each set of experiments, we fix a set of parameters and vary one. Unless otherwise stated, all of our experiments are repeated 5 times and the number reported is the average on all 5 repetitions.

Poisoning Rate In Fig. 1, we have 6 experiments where we fix the model to be logistic regression for all of them except one (Census random MLP) which uses a 5 layer perceptron with hidden layers sizes 32, 16 and 8. In all the experiments we set the number of shadow models to be 500 for census experiments and 1000 for Enron
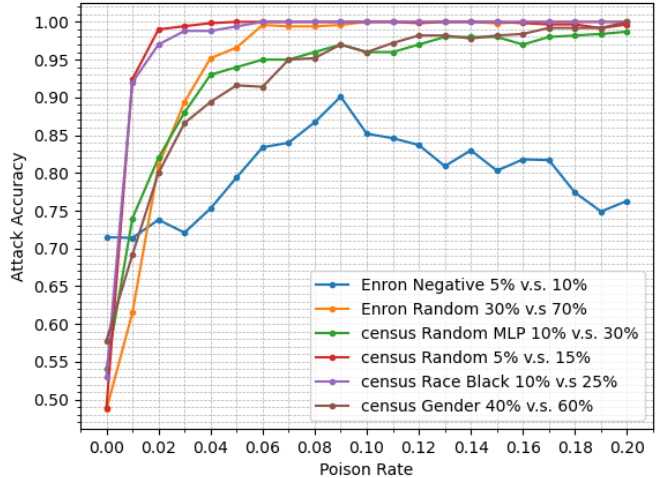

Fig. 1. Poison rate vs attack accuracy. We change the poisoning budget from 0% of the training size to 20%. Note that in most experiments, property inference without poisoning is not effective, but with less than 10% poisoning all of the experiments get accuracy above 90%. The curve marked as Census Random MLP shows the performance of our attack against MLP classifier on census dataset with random target feature. Also, observe that for one of the experiments (marked as Enron Negative), the accuracy of the attack starts decreasing after a certain level of poisoning, we discuss this phenomenon below.

experiments. The training set size is 1000 for Census experiments and 2000 for Enron experiments. We vary the poisoning rate from 0% to 20%. The number of black-box queries is set to 500 for experiments on Enron and 1000 for experiments on Census. The attack accuracy for all the target features is quite low when there is no poisoning. But with increase in poisoning rate, the attack accuracy dramatically improves and for all features, the accuracy reaches around 0.9 with just 10% poisoning.

The Enron negative sentiment experiment seems like an anomaly in Figure 1. However, the drop of accuracy with more poison points could be anticipated. We posit that for all features there is some point where adding more poisoning points will cause the accuracy to decrease. To understand this, one can think about the extreme case where 100% of the distribution is the poison data, which means there is no information about the clean distribution in the trained model and we would expect the attack to fail.

This effect is especially pronounced for properties that have very weak signal in the behavior of the final model. The Enron negative sentiment property produces the weakest signal among all the experiments because (1) the feature does not exist in the dataset and (2) it has the smallest difference in percentage ($t_1 - t_0$) among all the other experiments (5% vs 10%). We believe this is why it happens faster for the Sentiment property compared to other features. However, our insight suggests that this phenomenon should happen for any property for some poisoning ratio. To test this insight, we we tried various poisoning rates on the Enron dataset with random target feature. Figure 2 shows the result of this experiment where the accuracy of the attack starts to drop at poisoning rates around 30%. Interestingly, this phenomenon could be also explained using our Theorem as both ends of the range of certainty in the condition of our Theorem 9 will converge to ∞ when p approaches 1.

Number of Shadow Models The next set of experiments (See Fig 3) are to see the effect of the number of shadow models on the accuracy
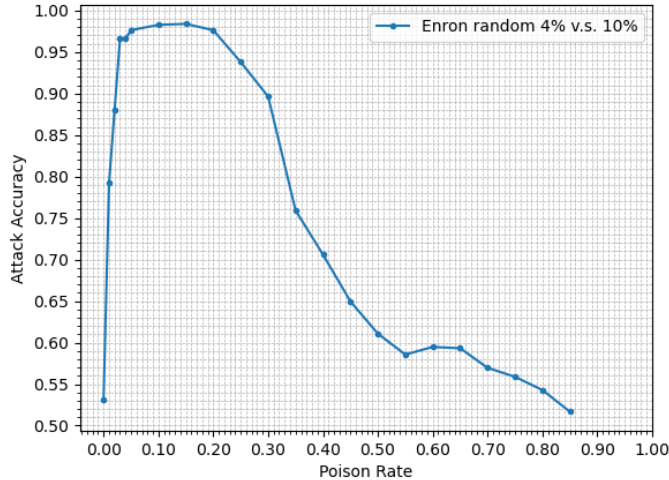
Fig. 2. Poison rate vs attack accuracy. This experiment shows that more poisoning becomes ineffective after a certain point and actually decreases the performance of the attack. The optimal poisoning ratio could be different across different experiments. This drop of attack accuracy was also observed in one of the experiments of Figure 1. Note that adversary can choose to use fewer poisoning points to ensure that the attack is optimal. In this experiment, the number of shadow models is 400 and the training set size is 1000.
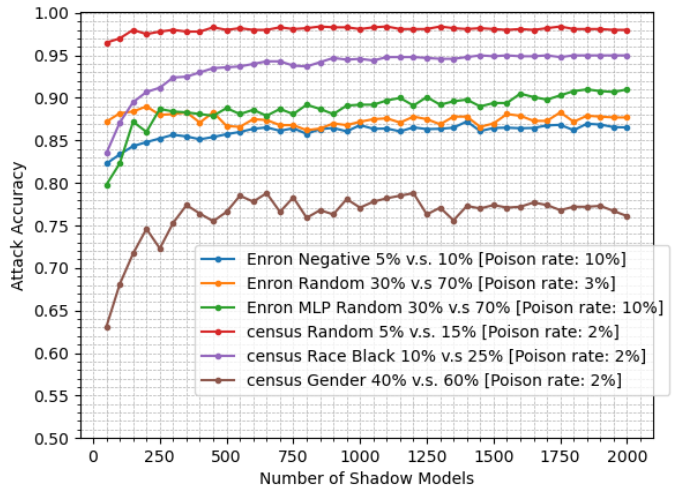

Fig. 3. Number of shadow models v.s. attack accuracy. Our attack used shadow model training to find important queries. This figure shows that in all experiments, less than 500 shadow models is enough to get the maximum accuracy. Note that in this experiment we also attack MLP classifier on Enron dataset for the random feature.

of the attack. For these experiments, we vary the number of shadow models from 50 to 2000. We notice that increasing the number of shadow models increases the attack accuracy and about 500 shadow models are sufficient to get close to maximum accuracy. Note that in this experiment we set the poisoning ratio to small values so that we can see the trend better. If larger poison ratio were chosen, in most experiments the attack reaches the maximum of 1 with very few shadow models and it is hard to see the trend. For instance, with 10% percent poisoning, the experiments with random feature (both census and Enron) would reach 100% accuracy with only 50 shadow models. This small number of shadow models makes the running time of the attack lower. In all of the experiments in this figure, the dataset size is set to 1000 except for the Enron negative sentiment experiment where the dataset size is 2000.

Training Set size In Fig. 4, we wanted to see the effect of training size on the effectiveness of the attack. Note that our theoretical attack suggests that larger training size should actually improve the attack because the models trained on larger training sets would have smaller generalization error and hence would be closer to a Bayes-optimal classifier. In Theorem 9 as the training set size increases, $\epsilon(n)$ and $\delta(n)$ will decrease which makes the attack more successful. In fact, our experiments verify this theoretical insight. In our experiments, we vary the training set size from 100 to 1500 and the upward trend is quite easy to observe. In this experiment we use 500 shadow models. Again, we have selected the poisoning rate and the number of shadow models in a way that the attack does not get accuracy 1.0 for small training sizes. In all experiments in this figure, the number of shadow models is set to 200. Also, the number of repetitions for this experiment is 2.

Undetectablilty of the Attack Recall that in our threat model, the adversary is able to poison a fraction of the training data. If the target model quality degrades significantly due to this poisoning, then it becomes easily detectable. Therefore, for the effectiveness of this attack, it is important that the quality of the model does not
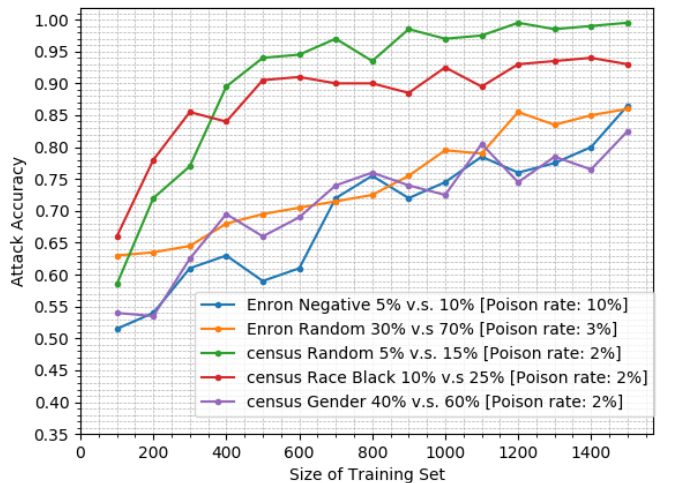

Fig. 4. Training set size v.s. attack accuracy. As was observed in our theoretical analysis, the size of the training set can impact the performance of our attack (because it affects the generalization error).
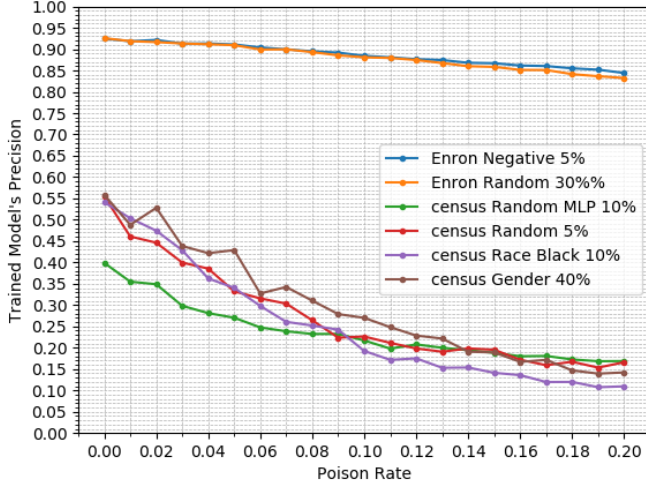
Fig. 5. Precision vs poison rate. Note that we did not balance the labels in our experiments and this is why we have precision smaller than 50% in the Census experiments.
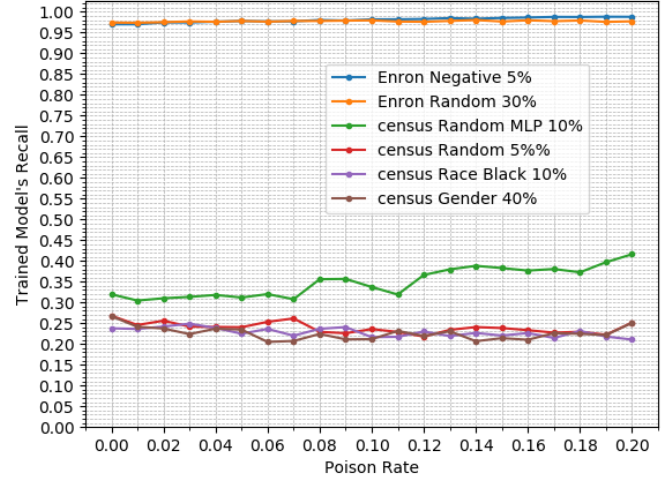

Fig. 6. Recall v.s. poison rate. We change the poison rate and capture the recall of resulting models in different experiments. Note that we did not use balanced label fractions in our experiments.

degrade[4]. We experimentally confirm that this is somewhat true with our poisoning strategy. See Fig 5 and Fig 6 for the precision and recall rate[5] for the model Logistic Regression where the poisoning rate varies from 0% to 20% for training set size of 1000. In general, the experiments show that the precision tends to decrease with a rather low slope and recall tend to increase by adding more poison data.The drop of precision and rise of recall can be explained by the fact that the poisoned model tends to predict label 1 more often than the non-poisoned model, because the poisoned data we add all has label 1. Note that for census data, the slope of decrease in precision is much higher than Enron. We believe this is because of the uncertain nature of Census dataset.However, it also worth mentioning that for all experiments in Census data, 4-5% poisoning is sufficient to get attack accuracy more than 90%. This means that, if one considers the drop in precision versus the attack accuracy, the census data is not much worse than enron. In our experiments, the size of the training set for Census experiments is set to 1000 and for Enron experiments the training set size is 2000.

Complexity of Target Models While in most of our experiments we fix the target model to be logistic regression (except for few experiments named Census MLP and Enron MLP in the figures),here we experiment with more complex architectures to see how our attack performs. We summarize the results in Table II. Based on our theoretical analysis, the effectiveness of the attack should depend on the model's performance in generalization to the training data distribution. Therefore, we expect the effectiveness of the attack to drop with more complex networks as the generalization error would increase when the number of parameters in the model increases. This might sound counter intuitive as the privacy problems are usually tied with over fitting and unintended memorization[7]. However, our experiments verify our theoretical intuition. We observe that as we add we more layers, the accuracy of the attack tends to drop. However, we

would expect this to change with larger training set sizes as the larger training size could compensate for the generalization error caused by higher numbers of parameters and overfitting. For instance, in the last row of Table II the accuracy increases significantly when we set the training size to 10000 and use more shadow models.

F. Comparison with WBAttack [17]

Since the work closest to ours is WBAttack, even though it is a white-box attack, we experimentally compare the performance of WBAttack to ours. WBAttack is an improved version of property inference attack of [16], where instead of a simple white-box shadow model training on the parameters of neural networks (which is done in [2]), they first try to decrease the entropy of the model parameters by sorting the neural network neurons according to the size of their input and output. They show that this reduction in randomness of the shadow models can increase the accuracy of attack, for the same number of shadow models. This attack is called the *vector* attack in [17]. In Table III, we see how our *black-box attack performance* compares with WBAttack. Notice that black-box with no poisoning (first 3 rows of the table) performs much worse that WBAttack on race and gender. However, WBAttack performs poorly on the random feature. In fact, the strength of our attack is to find a way to infer information about features similar to random that do not have high correlation with the label (and this is exactly what WBAttack is unable to attack), and do it in a fully black-box way. As we see in the columns below, with very small ratio of poisoning our attack get accuracy 1.0 on the random target property. It also beats the performance of WBAttack on other experiments with a very small number of poisoning points. Note that our attack also requires many fewer shadow models. For example with 10% poisoning, only 50 shadow models in our attack would beat the accuracy of WBAttack which uses 1000 shadow models. The small number of shadow models would be important in scenarios where the adversary does not have access to a lot of similar data. So in summary, our attack can improve on the performance WBAttack both in accuracy and number of shadow models, and of course in the access model which is black box. The cost of these improvements is allowing the adversary to choose a fraction of training set which is not an uncommon scenario in multi-party learning applications. To compare

---

[4]Note that we are only considering undetectability via black box access. In a eyes-on setting where the training data can be looked at, it is possible to have countermeasures that would detect poisoning by looking at the training data

[5]Precision: $\frac{\text{True positive}}{\text{True positive}+\text{False positive}}$, Recall: $\frac{\text{True positive}}{\text{True positive}+\text{False negative}}$

Architecture			Performance	
Hidden Layers	**Layer sizes**	**Training Size**	**Attack Accuracy**	**Shadow Models**
1	[2]	1000	1.0	600
2	[4 2]	1000	0.97	600
3	[8 4 2]	1000	0.94	600
4	[16 8 4 2]	1000	0.88	600
5	[32 16 8 4 2]	1000	0.81	600
5	[32 16 8 4 2]	10000	0.92	1000

TABLE II

COMPLEXITY OF TARGET MODELS VS ATTACK ACCURACY ON CENSUS DATA.

Experiment parameters		White-box Performance [17]		Black-box Performance		
Feature	**Training Size**	**# Shadow Models**	**Accuracy**	**# Shadow Models**	**Poison**	**Accuracy**
Census-Random	1000	1000	.52	1000	0	.5
Census-Gender	1000	1000	.96	1000	0	.61
Census-Race	1000	1000	.95	1000	0	.55
Census-Random	1000	1000	.52	100	0.05	1.0
Census-Gender	1000	1000	.96	100	0.03	.99
Census-Race	1000	1000	.95	100	0.05	.97
Census-Random	1000	1000	.52	50	0.1	1.0
Census-Gender	1000	1000	.96	50	0.1	1.0
Census-Race	1000	1000	.95	50	0.1	.98

TABLE III

COMPARISON WITH THE WHITE-BOX ATTACK OF [17]

the running time of the attacks, for the Census-Gender experiment in Table III, WBAttack ran in 1533s compared to our black-box attack that only took 161s on the same platform and execution environment. Note that we used the exact same size of held-out data for shadow models in both black-box and white-box experiments.

VIII. CONCLUSION AND FUTURE WORK

Poisoning attacks have emerged as a threat against security of machine learning. These poisoning attacks are usually studied in machine learning security where the goal of adversary is to increase the error or inject backdoor to the model. In this work, we initiated the study of poisoning adversaries that instead seek to the increase information leakage of trained models. We first presented a theoretical attack which succeeds against any Bayes optimal classifier. Then, we verified the effectiveness of out attack experimentally by implementing it and testing it against different datasets and classification tasks. Our results show that it does indeed succeed with very high probability. However, in our attack as well as in all the previous property attacks in the literature [3, 17], the attacker is assumed to have sample access to the underlying distributions D_- and D_+. It would be interesting to remove/relax this assumption in future work.

In this work, we tried to take a principled approach where experiments demonstrate the realistic threat of our attack, whereas the theoretical analysis gives intuition for the attacks. The theoretical analysis will hopefully also provide suggestion for where we might look in the future to either strengthen the attack or find mitigation. We leave this direction as future work.

REFERENCES

[1] *3 ways to train a secure machine learning model.* https://www.ericsson.com/en/blog/2020/2/training-a-machine-learning-model. Accessed: 2020-03-04.

[2] Giuseppe Ateniese et al. "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers". In: *arXiv preprint arXiv:1306.4447* (2013).

[3] Giuseppe Ateniese et al. "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers". In: *IJSN* 10.3 (2015), pp. 137–150. DOI: 10.1504/IJSN.2015.071829. URL: https://doi.org/10.1504/IJSN.2015.071829.

[4] Eugene Bagdasaryan et al. "How to backdoor federated learning". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2938–2948.

[5] Arjun Nitin Bhagoji et al. "Analyzing federated learning through an adversarial lens". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 634–643.

[6] Battista Biggio, Blaine Nelson, and Pavel Laskov. "Poisoning attacks against support vector machines". In: *arXiv preprint arXiv:1206.6389* (2012).

[7] Nicholas Carlini et al. "The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks". In: *USENIX Security Symposium*. 2018.

[8] Xinyun Chen et al. *Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning*. 2017. arXiv: 1712.05526 [cs.CR].

[9] Christopher A Choquette Choo et al. "Label-Only Membership Inference Attacks". In: *arXiv preprint arXiv:2007.14321* (2020).

[10] Ilias Diakonikolas et al. "Robust estimators in high-dimensions without the computational intractability". In: *SIAM Journal on Computing* 48.2 (2019), pp. 742–864.

[11] Ilias Diakonikolas et al. "Sever: A robust meta-algorithm for stochastic optimization". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 1596–1606.

[12] Dimitrios I Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. "Lower bounds for adversarially robust pac learning". In: *arXiv preprint arXiv:1906.05815* (2019).

[13] Cynthia Dwork et al. "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. Lecture Notes in Computer Science. Springer, 2006, pp. 265–284. DOI: 10.1007/11681878_14. URL: https://doi.org/10.1007/11681878%5C_14.

[14] Andrew Frank, Arthur Asuncion, et al. "UCI machine learning repository, 2010". In: *URL http://archive. ics. uci. edu/ml* 15 (2011), p. 22.

[15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. "Model inversion attacks that exploit confidence information and basic countermeasures". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015, pp. 1322–1333.

[16] Karan Ganju et al. "Property Inference Attacks on Fully Connected Neural Networks Using Permutation Invariant Representations". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS '18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 619–633. ISBN: 9781450356930. DOI: 10.1145/3243734.3243834. URL: https://doi.org/10.1145/3243734.3243834.

[17] Karan Ganju et al. "Property Inference Attacks on Fully Connected Neural Networks using Permutation Invariant Representations". In: *CCS '18*. 2018.

[18] Sanjam Garg et al. "Obliviousness Makes Poisoning Adversaries Weaker". In: *arXiv preprint arXiv:2003.12020* (2020).

[19] Micah Goldblum et al. "Data Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses". In: *arXiv preprint arXiv:2012.10544* (2020).

[20] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[21] Yingzhe He et al. *Towards Privacy and Security of Deep Learning Systems: A Survey*. 2019. arXiv: 1911.12562 [cs.CR].

[22] Andrew Ilyas et al. "Adversarial examples are not bugs, they are features". In: *arXiv preprint arXiv:1905.02175* (2019).

[23] Matthew Jagielski. "Subpopulation Data Poisoning Attacks". In: 2019.

[24] Matthew Jagielski, Jonathan Ullman, and Alina Oprea. "Auditing Differentially Private Machine Learning: How Private is Private SGD?" In: *Advances in Neural Information Processing Systems* 33 (2020).

[25] Bargav Jayaraman and David Evans. "Evaluating differentially private machine learning in practice". In: *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 2019, pp. 1895–1912.

[26] Bryan Klimt and Yiming Yang. *The enron corpus: A new dataset for email classification research*. 2004.

[27] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. "Stronger data poisoning attacks break data sanitization defenses". In: *arXiv preprint arXiv:1811.00741* (2018).

[28] Kevin A Lai, Anup B Rao, and Santosh Vempala. "Agnostic estimation of mean and covariance". In: *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2016, pp. 665–674.

[29] Aleksander Madry et al. "Towards deep learning models resistant to adversarial attacks". In: *arXiv preprint arXiv:1706.06083* (2017).

[30] Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. "Learning under p-Tampering Attacks". In: *Algorithmic Learning Theory*. PMLR. 2018, pp. 572–596.

[31] Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. "Learning under p-tampering poisoning attacks". In: *Annals of Mathematics and Artificial Intelligence* 88.7 (2020), pp. 759–792.

[32] Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. "The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 4536–4543.

[33] Saeed Mahloujifar and Mohammad Mahmoody. "Block-wise p-tampering attacks on cryptographic primitives, extractors, and learners". In: *Theory of Cryptography Conference*. Springer. 2017, pp. 245–279.

[34] Saeed Mahloujifar and Mohammad Mahmoody. "Can Adversarially Robust Learning LeverageComputational Hardness?" In: *Algorithmic Learning Theory*. PMLR. 2019, pp. 581–609.

[35] Saeed Mahloujifar, Mohammad Mahmoody, and Ameer Mohammed. "Data poisoning attacks in multi-party learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 4274–4283.

[36] Luca Melis et al. "Exploiting Unintended Feature Leakage in Collaborative Learning". In: *2019 IEEE Symposium on Security and Privacy (SP)* (May 2019). DOI: 10.1109/sp.2019.00029. URL: http://dx.doi.org/10.1109/SP.2019.00029.

[37] Milad Nasr, Reza Shokri, and Amir Houmansadr. "Comprehensive privacy analysis of deep learning". In: *2019 ieee symposium on security and privacy*. 2019.

[38] Milad Nasr, Reza Shokri, and Amir Houmansadr. "Machine learning with membership privacy using adversar-

ial regularization". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 634–646.

[39] Adarsh Prasad et al. "Robust estimation via robust gradient estimation". In: *arXiv preprint arXiv:1802.06485* (2018).

[40] *Privacy-preserving Collaborative Machine Learning*. https://medium.com/sap-machine-learning-research/privacy-preserving-collaborative-machine-learning-35236870cd43. Accessed: 2020-03-04.

[41] Alexandre Sablayrolles et al. "White-box vs Black-box: Bayes Optimal Strategies for Membership Inference". In: *ArXiv abs/1908.11229* (2019).

[42] Ali Shafahi et al. "Poison frogs! targeted clean-label poisoning attacks on neural networks". In: *Advances in Neural Information Processing Systems*. 2018, pp. 6103–6113.

[43] Reza Shokri et al. "Membership inference attacks against machine learning models". In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, pp. 3–18.

[44] Congzheng Song and Vitaly Shmatikov. "Auditing data provenance in text-generation models". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 196–206.

[45] Liwei Song and Prateek Mittal. "Systematic Evaluation of Privacy Risks of Machine Learning Models". In: *arXiv preprint arXiv:2003.10595* (2020).

[46] Liwei Song, Reza Shokri, and Prateek Mittal. "Privacy risks of securing machine learning models against adversarial examples". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 241–257.

[47] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. "Certified defenses for data poisoning attacks". In: *Advances in neural information processing systems*. 2017, pp. 3517–3529.

[48] Fnu Suya et al. "Model-Targeted Poisoning Attacks: Provable Convergence and Certified Bounds". In: *arXiv preprint arXiv:2006.16469* (2020).

[49] Binghui Wang and Neil Zhenqiang Gong. "Stealing hyperparameters in machine learning". In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 36–52.

APPENDIX

In this section we prove the tools we used to prove our main Theorem. Specifically, We prove Claim 12, Corollary 13 and Claim 14.

Proof of Claim 12. We have

$$
\Pr[\tilde{Y} = 1 \mid \tilde{X} = x]
$$
$$
= \Pr[\tilde{Y} = 1 \mid \tilde{X} = x \wedge E] \cdot \Pr[E \mid \tilde{X} = x]
$$
$$
+ \Pr[\tilde{Y} = 1 \mid \tilde{X} = x \wedge \bar{E}] \cdot \Pr[\bar{E} \mid \tilde{X} = x]
$$
$$
= \Pr[E \mid \tilde{X} = x] + \Pr[Y = 1 \mid X = x] \cdot \Pr[\bar{E} \mid \tilde{X} = x] \quad (1)
$$

Now we should calculate the probability $\Pr[E \mid \tilde{X} = x]$ to get the exact probabilities. We have

$$
\Pr[E \mid \tilde{X} = x]
$$
$$
= \frac{\Pr[\tilde{X} = x \mid E] \cdot \Pr[E]}{\Pr[\tilde{X} = x \mid E] \cdot \Pr[E] + \Pr[\tilde{X} = x \mid \bar{E}] \cdot \Pr[\bar{E}]}
$$
$$
= \frac{\Pr[X_+ = x] \cdot p}{\Pr[X_+ = x] \cdot p + \Pr[X = x] \cdot (1 - p)} \quad (2)
$$

Now for all $x \in \mathcal{X}$ such that $f(x) = 1$ we have

$$
\Pr[X = x] = t \cdot \Pr[X_+ = x]. \quad (3)
$$

and for all $x \in \mathcal{X}$ such that $f(x) = 0$ we have

Now combining Equations 2 and 3, for all $x \in \mathcal{X}$ such that $f(x) = 1$ we have

$$
\Pr[E \mid \tilde{X} = x] = \frac{p}{p + t \cdot (1 - p)}. \quad (4)
$$

Combining equations 1 and 4 we get

$$
\Pr[\tilde{Y} = 1 \mid \tilde{X} = x] = \frac{p}{p + t(1 - p)}
$$
$$
+ \frac{t(1 - p)}{p + t(1 - p)} \cdot \Pr[Y = 1 \mid X = x]
$$

which finishes the proof. $\square$

Proof of Corollary 13. if $\mathsf{crt}(x) \leq \frac{p - 2\tau \cdot t}{t(1 - p)}$ then by Claim 12 we have

$$
\Pr[\tilde{Y} = 1 \mid \tilde{X} = x]
$$
$$
= \frac{p}{p + t(1 - p)} + \frac{t(1 - p)}{p + t(1 - p)} \cdot \Pr[Y = 1 \mid X = x]
$$
$$
= \frac{p}{p + t(1 - p)} + \frac{t(1 - p)}{p + t(1 - p)} \cdot \left(\frac{1 - \mathsf{crt}(x)}{2}\right)
$$
$$
\geq \frac{p}{p + t(1 - p)} + \frac{t(1 - p)}{p + t(1 - p)} \cdot \left(\frac{t(1 - p) - p + 2t\tau}{2t(1 - p)}\right)
$$
$$
= \frac{t(1 - p) + p + 2\tau t}{2(p + t(1 - p))}
$$
$$
= \frac{1}{2} + \frac{\tau t}{p + t(1 - p)}.
$$

To show the other direction, we can follow the exact same steps in the opposite order. $\square$

Proof of Claim 14. For all $x \in \mathcal{X}$ such that $C_\tau(x) = 1$, using Corollary 13, if $t = t_1$ then we have

$$
\Pr[\tilde{Y} = 1 \mid \tilde{X} = x] < 0.5 - \frac{\tau \cdot t_1}{p + (1 - p) \cdot t_1} \quad (5)
$$

and if $t = t_0$ then

$$
\Pr[\tilde{Y} = 1 \mid \tilde{X} = x] \geq 0.5 + \frac{\tau \cdot t_0}{p + (1 - p) \cdot t_0} \quad (6)
$$

This implies that for both cases of $t = t_0$ and $t = t_1$ we have

$$
|\mathsf{crt}(x, \tilde{D})| \geq \frac{2\tau t}{p + (1 - p) \cdot t} \quad (7)
$$

And it also implies that for case of $t = t_0$ we have

$$
h^*(x, \tilde{D}) = 1 \quad (8)
$$

and for $t = t_1$ we have

$$
h^*(x, \tilde{D}) = 0. \quad (9)
$$

Now we state a lemma that will be used in the rest of the proof:

Lemma 15. *For any distribution $D \equiv (X, Y)$ where $\mathrm{Supp}(Y) = \{0,1\}$ and any classifier $h : \mathrm{Supp}(X) \to \{0,1\}$ we have:*

$$\mathsf{Risk}(h, D) = \mathsf{Bayes}(D) + \mathop{\mathbf{E}}_{x \leftarrow X}\left[|\tilde{h}x) - h^*(x, D)| \cdot |\mathsf{crt}(x, D)|\right]$$

where h^* is the Bayes-Optimal classifier as defined in Definition 5.

Proof. For simplicity, in this proof we use $h^*(x)$ instead of $h^*(x, D)$. We have

$\mathsf{Risk}(h, D)$

$= \mathop{\mathbf{E}}_{(x,y) \leftarrow (X,Y)}[\tilde{h}(x) \neq y]$

$= \mathop{\mathbf{E}}_{(x,y) \leftarrow (X,Y)}[h^*(x) \neq y \mid \tilde{h}(x) = h^*(x)] \Pr[\tilde{h}(X) = h^*(X)]$

$+ \mathop{\mathbf{E}}_{(x,y) \leftarrow (X,Y)}[h^*(x) = y \mid \tilde{h}(x) \neq h^*(x)] \Pr[\tilde{h}(X) \neq h^*(X)]$

$= \mathsf{Bayes}(D)$

$- \mathop{\mathbf{E}}_{(x,y) \leftarrow (X,Y)}[h^*(x) \neq y \mid \tilde{h}(x) \neq h^*(x)] \Pr[\tilde{h}(X) \neq h^*(X)]$

$+ \mathop{\mathbf{E}}_{(x,y) \leftarrow (X,Y)}[h^*(x) = y \mid \tilde{h}(x) \neq h^*(x)] \Pr[\tilde{h}(X) \neq h^*(X)]$

$= \mathsf{Bayes}(D)$

$- \mathop{\mathbf{E}}_{(x,y) \leftarrow (X,Y)}[|y - h^*(x)||\tilde{h}(x) - h^*(x)|]$

$+ \mathop{\mathbf{E}}_{(x,y) \leftarrow (X,Y)}[(1 - |y - h^*(x)|)|\tilde{h}(x) - h^*(x)|]$

$= \mathsf{Bayes}(D) + \mathop{\mathbf{E}}_{(x,y) \leftarrow (X,Y)}[(1 - 2|y - h^*(x)|)|\tilde{h}(x) - h^*(x)|]$

$= \mathsf{Bayes}(D) + \mathop{\mathbf{E}}_{x \leftarrow X}[|\tilde{h}(x) - h^*(x)| \mathop{\mathbf{E}}_{y \leftarrow Y|X=x}[1 - 2|y - h^*(x)|]]$

Now we show that $\mathbf{E}_{y \leftarrow Y|X=x}[1 - 2|y - h^*(x)|] = |\mathsf{crt}(x)|$. The reason is that, if $\Pr[Y = 1|X = x] \geq 0.5$ then $h^*(x) = 1$ and we have $\mathbf{E}_{y \leftarrow Y|X=x}[1 - 2|y - h^*(x)|] = 2\,\mathbf{E}_{y \leftarrow Y|X=x}[y] - 1 = |\mathsf{crt}(x)|$. And if $\Pr[Y = 1|X = x] < 0.5$ then $h^*(x) = 0$ and we have $\mathbf{E}_{y \leftarrow Y|X=x}[1 - 2|y - h^*(x)|] = 1 - 2\,\mathbf{E}_{y \leftarrow Y|X=x}[y] = |\mathsf{crt}(x)|$. Therefore, the proof is complete. □

Now we are ready to complete the proof. Assume that we have $t = t_0$ and

$$\Pr_{x \leftarrow X|C_\tau(x)=1}[\tilde{h}(x) = 1] < 0.5 + \gamma \tag{10}$$

let $\alpha = \mathsf{Risk}(\tilde{h}, D) - \mathsf{Bayes}(D)$. By Lemma 15 we have

$\alpha = \mathop{\mathbf{E}}_{x \leftarrow \tilde{X}}\left[|\tilde{h}(x) - h^*(x, \tilde{D})| \cdot |\mathsf{crt}(x, \tilde{D})|\right]$

$\geq \mathop{\mathbf{E}}_{x \leftarrow \tilde{X}}\left[|\tilde{h}x) - h^*(x, \tilde{D})| \cdot |\mathsf{crt}(x, \tilde{D})| \mid C_\tau(x) = 1\right] \cdot \Pr[C_\tau(\tilde{X}) = 1]$

$\geq (\mathop{\mathbf{E}}_{x \leftarrow \tilde{X}|C_\tau(\tilde{X})=1}[1 - \tilde{h}(x)]) \cdot \frac{2\tau t_0}{p + (1 - p) \cdot t_0} \cdot \Pr[C_\tau(\tilde{X}) = 1]. \tag{11}$

Note that the last line above is derived by combining Equations 7 and 8. Now using our assumption in Equation 10 we have

$$\alpha > \frac{2(0.5 - \gamma)\tau \cdot t_0}{(p + (1 - p)t_0)} \cdot \Pr[C_\tau(\tilde{X}) = 1]$$

Now we note that for any x such that $f(x) = 1$ we have

$$\Pr[\tilde{X} = x] = \frac{p + (1 - p)t}{t} \cdot \Pr[X = x])$$

because of the way poisoning is done[6]. Therefore we have

$$\Pr[C_\tau(\tilde{X}) = 1] = \frac{p + (1 - p)t}{t} \cdot \Pr[C_\tau(X) = 1]) \tag{12}$$

and we get

$\alpha > \frac{2(0.5 - \gamma)\tau \cdot t_0}{(p + (1 - p)t_0)} \cdot \frac{p + (1 - p)t_0}{t_0} \cdot \Pr[C_\tau(X) = 1])$

$= (1 - 2\gamma) \cdot \tau \cdot \Pr[C_\tau(X) = 1])$

$\geq (1 - 2\gamma) \cdot \tau \frac{\epsilon(n)}{\tau(1 - 2\gamma)} = \epsilon(n).$

This means that if the assumption of Equation 10 holds then the error would be larger than $\epsilon(n) + \mathsf{Bayes}(\tilde{D})$ which means the probability of 10 happening is at most $\delta(n)$ by Bayes-optimality of the learning algorithm. Namely,

$$\Pr_{\substack{S \leftarrow \tilde{D}^n \\ \tilde{h} \leftarrow L(S)}}\left[\Pr_{x \leftarrow X|C_\tau(x)=1}\left[\tilde{h}(x) = 1\right] \geq 0.5 + \gamma\right] \geq 1 - \delta(n).$$

In case of $t = t_1$, the proof is similar. We first assume

$$\Pr_{x \leftarrow X|C_\tau(x)=1}[\tilde{h}(x) = 0] > 0.5 - \gamma \tag{13}$$

Using Equations 7 and 9 we get the following (similar to Inequality 11 for $t = t_0$)

$$\alpha > (\mathop{\mathbf{E}}_{x \leftarrow \tilde{X}|C_\tau(\tilde{X})=1}[\tilde{h}(x)]) \cdot \frac{2\tau t_1}{p + (1 - p) \cdot t_1} \cdot \Pr[C_\tau(\tilde{X}) = 1]. \tag{14}$$

Therefore, using Equations 12 and 13 we get

$$\alpha > \epsilon(n)$$

which implies

$$\Pr_{\substack{S \leftarrow \tilde{D}^n \\ \tilde{h} \leftarrow L(S)}}\left[\Pr_{x \leftarrow X|C_\tau(x)=1}\left[\tilde{h}(x) = 1\right] \leq 0.5 - \gamma\right] \geq 1 - \delta(n).$$

□

---

[6]For simplicity we are assuming the support of distribution is discrete. Otherwise, we have to have the same argument for measurable subsets instead of individual instances.