Learning under p-Tampering Attacks

Saeed Mahloujifar[*] and **Dimitrios I. Diochnos**[†] and **Mohammad Mahmoody**[‡]

Abstract

Recently, Mahloujifar and Mahmoody (TCC'17) studied attacks against learning algorithms using a special case of Valiant's malicious noise, called p-tampering, in which the adversary gets to change any training example with independent probability p but is limited to choose 'adversarial' examples only with correct labels. They obtained p-tampering attacks that increase the error probability in the so called 'targeted' poisoning model in which the adversary's goal is to increase the loss of the trained hypothesis over a particular test example. At the heart of their attack was an efficient algorithm to bias the average of any bounded real-valued function through p-tampering.

In this work, we present new improved biasing attacks for biasing the average output of bounded real-valued functions. Our new biasing attacks achieve in *polynomial-time* the best bias achieved by MM16 through an *exponential* time p-tampering attack. Our improved biasing attacks, directly imply improved p-tampering attacks against learners in the targeted poisoning model. As a bonus, our improved attacks come also with much simpler analysis compared to previous attacks. We also study the possibility of PAC learning under p-tampering attacks in the *non-targeted* (aka indiscriminate) setting where the adversary's goal is to increase the risk of the hypothesis (for a *random* test example). We show that PAC learning is *possible* under p-tampering poisoning attacks basically whenever it is possible in the realizable setting without attacks.

Introduction

In his seminal work [39], Valiant introduced the Probably Approximately Correct (PAC) model of learning that triggered a significant amount of work in the theory of machine learning.[1] An important characteristic of learning algorithms is their ability to cope with noise. Valiant did also initiate a study of adversarial noise [40] in which incoming training examples are chosen, with independent probability p, by an adversary. As a result, this type of noise is called *malicious*. Subsequently, Kearns and Li [24] proved impossibility of PAC learning under malicious noise by relying on the existence of *mistakes* (i.e., wrong labels) in adversarial examples for a carefully chosen initial distribution.

Bshouty, et al. [11] studied a closely related model in which the adversary is allowed to make its choices based on the full knowledge of the original training examples. While the results of [24] make use of particular pathological distributions from which the malicious samples are drawn[2], in this work we are interested in studying attackers against learners in a setting where the attackers do *not* have any control over the the original distributions, but they can choose the malicious distribution in certain (still restricted) ways.

Poisoning attacks. Impossibility results against learning under adversarial noise could be seen as attacks against learners in which the attacker injects some malicious training examples to the training set and tries to prevent the learner from finding a hypothesis with low risk. Such attackers, in general, are studied in the context of *poisoning* (a.k.a causative) attacks [4, 36, 42] in which an adversary aims at directing a learner towards generating a hypothesis that performs badly during the test phase.[3] Such attacks could happen naturally when a learning process happens over time [33, 34] and the adversary has some noticeable chance of injecting or sub-

[*]University of Virginia, saeed@virginia.edu. Supported by UVa's SEAS Research Innovation Award.

[†]University of Virginia, diochnos@virginia.edu.

[‡]University of Virginia, mohammad@virginia.edu. Supported by NSF CAREER award CCF-1350939, and UVa's SEAS Research Innovation Award.

[1]The original model studies learnability in a distribution-free sense, it also make sense for classes of distributions; [6].

[2]This is similar to [9, 19] that deals with the sample complexity of distribution-free learning.

[3]At a technical level, the malicious noise model also allows the adversary to know the *full* state (and thus the private randomness) of the learner, while this knowledge is not given to the adversary of the poisoning attacks, who might be limited in various other ways as well.

stituting malicious training data in an online manner. A stronger form of poisoning attacks are the so called *targeted* (poisoning) attacks [36], where the adversary performs the poisoning attack while she has a particular test example in mind, and her goal is to make the final generated hypothesis fail on that particular test example. While poisoning attacks against *specific* learners were studied before [4, 36, 42], the recent work of Mahloujifar and Mahmoody [27] presented a generic *black-box* targeted poisoning attack that could adapt to apply to *any learner*, so long as there is an initial non-negligible error over the target point.

p-tampering attacks. The work of [27] proved their result using a special case of Valiant's malicious noise, called p-tampering, in which the attacker can only use *mistake-free malicious noise*. Namely, similar to Valiant's model, any incoming training example might be chosen adversarially with independent probability p (see Definition 5 for a formalization). The difference between p-tampering noise and Valiant's adversarial noise (and even from all of its special cases studied before [37]) is that whenever the p-tampering adversary is allowed to tamper with a particular example, it can only choose *valid* tampered examples (to substitute the original examples) that have *correct* labels.[4] As such, although the attributes can change pretty much arbitrarily in the tampered examples, the label of the tampered examples shall reflect the correct label[5]. Therefore, as opposed to the general model of Valiant's malicious noise, p-tampering noise/attacks are 'defensible' as the adversary can always claim that a malicious training example is indeed generated from the same original distribution from which the rest of the training examples are generated. Similar notions of defensible attacks are previously explored in cryptography [2, 23].

Poisoning through biasing. At the heart of the poisoning attacks of [27] against learners was a basic p-tampering attack for *biasing* the average output of bounded real-valued functions. In particular, [27] proved that for any (efficient) function f mapping inputs drawn from distributions like $S \equiv D^n$ (consisting of n iid 'blocks') to $[0, 1]$, there is always an (efficient) p-tampering attacker A who changes the input distribution S into $\widehat{S}$ while increasing the average of the output by at least $\frac{2p}{3+4p} \cdot \mathrm{Var}[f(S)]$ where $\mathrm{Var}[\cdot]$ is the variance.[6] For the special case of *Boolean* function $f(\cdot)$, or when the p-tampering attacker could be *exponential time*, they could achieve a better bias of $\frac{p}{1+p\cdot\mu-p} \cdot \mathrm{Var}[f(S)]$ where $\mu = \mathbf{E}[f(S)]$ is the original average of $f(S)$. After obtaining biasing attacks, [27] derived their p-tampering targeted poisoning attacks from them by essentially biasing the average of the loss function $\mathrm{Loss}(h(x), y)$ where h is the learned hypothesis and $(x, y) = d$ is the target test.

Relation to Robustness. The robustness of a learner [21, 43, 45] refers to its behavior when the test test are close, but not necessarily drawn from the same distribution. The question in that setting is how well the learned hypothesis performs on the test set. Learning under p-tampering can be seen as a generalization of algorithmic robustness in which the training distribution can *adaptively* and *adversarially* deviate form the testing distribution without using wrong labels.

Comparison with evasion attacks. In the last few years neural network based architectures explored the so-called *adversarial perturbations* for some correctly classified instances so that the perturbed instances are misclassified [38]. Such resulting misclassified perturbed instances are called *adversarial examples* and attacks aimed at finding such examples are called *evasion attacks* [8, 13, 22, 28, 30, 44]. The goal of evasion attacks is quite different from poisoning attacks: in poisoning attacks the tampering happens over the training data; in evasion attacks no tampering to the training data is allowed, but it is allowed for the test example.

Our Results

Improved p-tampering biasing attacks. Our main technical result in this work is to improve the efficient (polynomial-time) p-tampering biasing attack of [27] to achieve the bias of $\frac{p}{1+p\cdot\mu-p} \cdot \mathrm{Var}[f(S)]$ (where $\mu = \mathbf{E}[f(S)]$ for $S \equiv D^n$ and $\mathrm{Var}[\cdot]$ is the variance) in *polynomial time* and for *real-valued* bounded functions with output in $[0, 1]$ (see Theorem 1). This main result immediately allows us to get improved polynomial-time targeted p-tampering attacks against learners for scenarios where the loss function is not Boolean (see Corollary 2). As in [27], our attacks are black-box and apply to any learning problem P and any learner L for P as long as L has 'noticeable' error over a test example d.

Special case of p-resetting attacks. The biasing attack of [27] has an extra property: for each block (or training example) d_i, if the adversary gets to tamper with d_i, it either does not change d_i at all, or it simply 'resets' it by re-sampling it from the training distribution D. In this work, we refer to such limited forms of p-tampering attacks as *p-resetting* attacks. Interesingly, p-resetting attacks were previously studied in the work of Bentov, Gabizon, and Zuckerman [7] in the context of (ruling out) extracting uniform randomness from Bitcoin's blockchain [29] when the adversary controls p fraction of the computing power, and thus

[4]This is assuming that the original training distribution only contains correct labels.

[5] For example, the adversary can repeatedly present the same example to the learner, thus reducing the effective sample size, or it can be the case that the adversary returns correct examples that are chosen against the learner's algorithm and based on the whole history of the examples so far.

[6]Note that the bias shall somehow depend on $\mathrm{Var}[f(S)]$ since constant functions cannot be biased.

it has the chance p of obtaining the next block, which she can discard/reset.[7] [7] showed how to achieve bias $p/12$ when the original (untampered) distribution D is uniform and the function f is Boolean and balanced.[8] As a special case of p-tampering attacks, p-resetting attacks have interesting properties that are not present in general p-tampering attacks. For example, if the original training distribution D includes wrong labels with probability ε, this probability will only go up to at most $(1+p)\cdot\varepsilon = \varepsilon + p\cdot\varepsilon$ under a p-resetting attack, while it could go up to $\varepsilon + p$ under p tampering attacks. Motivated by special applications of p resetting attacks and the special properties of p-resetting attacks, in this work we also study such attacks over arbitrary block distributions D and achieve bias of at least $\frac{p}{1+p\cdot\mu}\cdot\mathrm{Var}[f(S)]$, improving upon the previous bias of $\frac{2p}{3+4p}\cdot\mathrm{Var}[f(S)]$ proved in [27].

PAC learning under non-targeted poisoning. We also study the power of p-tampering (and p-resetting) attacks in the *non-targeted* setting where the adversary's goal is simply to increase the risk of the generated hypothesis.[9] In this setting, it is indeed meaningful to study the possibility (or impossibility) of PAC learning, as the test example is chosen at random. We show that in this model, p-tampering attacks cannot prevent PAC learnability in 'realizable' settings where there is always a hypothesis consistent with the training data (see Theorem 10).

Bounded budget attackers. In the full version of this paper [26], we go beyond p-tampering attacks and study PAC learning under more powerful adversaries who might *choose* the training examples that are tampered with but are still limited to choose $\leq p\cdot n$ such examples. We show that PAC learning under such adversaries depends on whether the adversary makes its tampering choices *before* or *after* getting to see the original 'honest' sample d_i. We call these two classes of attacks strong/weak p-budget tampering attacks. Our notions of p-budget tampering are inspired by notions of (strong) adaptive corruption [12, 20] in cryptographic context. Our impossibility result for PAC learnability under strong p-budget attacks shows that PAC learning under 'mistake-free' adversarial noise is *not* always possible.

Applications beyond attacking learners. Similar to how [27] used their biasing attacks in applications other than attacking learners, our new biasing attacks can also be used to obtain improved polynomial-time attacks for biasing the output bit of any seedless randomness extractors [14, 35, 41], as well as blockwise p-tampering (and p-resetting) attacks against security of certain cryptographic primitives (e.g., encryption, secure computation, etc.). As in [27], our new improved biasing attacks apply to any *joint* distribution (e.g., a martingale). In this work, however, we focus on the case of product distributions that already includes all the main applications to learning and include all the main ideas even for the general case of random processes. See the work of [27] for such applications.

Ideas behind Our Biasing Attacks. At a high level, the attacks of [27] were simple to describe, while their analyses were extremely complicated and heavily relied on carefully chosen potential functions based on ideas from [3] in which authors presented a p-tampering biasing attack for the special case of uniform Boolean blocks (i.e., $D\equiv U_1$). Our new (polynomial time) attacks use completely different ideas as they have a more complicated description, while the analysis of our attacks are indeed much simpler.

Our new biasing attacks build upon ideas developed in previous work [5,7,17,18,32] in the context of attacking deterministic randomness extractors from Santha-Vazirani sources [35]. In [27] the authors generalized the idea of 'half-space' sources (introduced in [17, 32]) to real-valued functions, using which it was shown how to find p-tampering biasing attacks with same bias $\frac{p}{1+p\cdot\mu-p}\cdot\mathrm{Var}[f(S)]$ as ours using inefficient *exponential* time attacks. Achieving the same bias *efficiently*, however, is the main technical contribution.

More formally, let $d_{\leq i} = (d_1,\ldots,d_i)$ be the first i blocks given as input to a function f (or alternatively the first i training examples, when we attack learners). Note that some of the blocks in $(d_1,\ldots,d_i)$ might be the result of previous tamperings. Now, suppose the adversary gets the chance to determine a new value d'_i for d_i in its p-tampering attack (which happens with probability p) knowing only the previously generated blocks $(d_1,\ldots,d_{i-1})$. In [27] it was shown that there always exists some d'_i (that could be found in *exponential* time) such that choosing it will lead to the bias $\frac{p}{1+p\cdot\mu-p}\cdot\mathrm{Var}[f(S)]$. They also showed how to choose d'_i efficiently, but that resulted in achieving smaller bias of $\frac{2p}{3+4p}\cdot\mathrm{Var}[f(S)]$. The analysis of the efficient attacks of [27] relies on potential functions involving 'partial averages' of $f(\cdot)$ defined as

$$\hat{f}[d_{\leq i}] = \mathop{\mathbf{E}}_{d_{i+1},\ldots,d_n\leftarrow D^{n-i}}[f(d_1,\ldots,d_n)].$$

One of the key ideas enabling the attacks of this work is to design our attacks' *algorithms* (and not their analyses) directly based on the (unrealistic) assumption that we have access to an oracle providing the partial averages $\hat{f}[d_{\leq i}]$ of $f(\cdot)$. By leveraging on the oracle $\hat{f}[d_{\leq i}]$

[7] To compare the terminologies, the work of [7] studies *p-resettable* sources of randomness, while here we study p-resetting attackers that generate such sources.

[8] The running time of the p-resetting attacker of [7] was $\mathrm{poly}(n, 2^{|D|})$ where $|D|$ is the length of the binary representation of any $d\leftarrow D$. In contrast, our p-resetting attacks run in time $\mathrm{poly}(n,|D|)$.

[9] In the targeted setting, the ε parameter of (ε,δ)-PAC learning goes away, due to the pre-selection of the target test.

we design our attacks in a way that we can compute their achieved biases *exactly* (rather than bounding them using potential functions as it was done in [27]). Fortunately, although the partial averages $\hat{f}[d_{\leq i}]$ are not *exactly* computable in polynomial time, they can indeed be efficiently approximated within arbitrary small additive error. As we show, our attacks are also robust to such approximation, and by using the approximations of $\hat{f}[d_{\leq i}]$ (rather than their exact values) we can still control how much bias is achieved.

Preliminaries

Notation. We use calligraphic letters (e.g., $\mathcal{D}$) for sets and capital non-calligraphic letters (e.g., D) for distributions. By $d \leftarrow D$ we denote that d is sampled from D. For a randomized algorithm $L(\cdot)$, by $y \leftarrow L(x)$ we denote the randomized execution of L on input x outputting y. For joint distributions (X, Y), by $(X \mid y)$ we denote the conditional distribution $(X \mid Y = y)$. By $\mathrm{Supp}(D) = \{d \mid \Pr[D = d] > 0\}$ we denote the support set of D. By $T^D(\cdot)$ we denote an algorithm $T(\cdot)$ with oracle access to a sampler for D. By $D \equiv G$ we denote that distributions D, G are identically distributed. By D^n we denote n iid samples from D. By $\varepsilon(n) \leq \frac{1}{\mathrm{poly}(n)}$ we mean $\varepsilon(n) \leq \frac{1}{n^{\Omega(1)}}$ and by $t(n) \leq \mathrm{poly}(n)$ we mean $t(n) \leq n^{O(1)}$.

A learning problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \mathrm{Loss})$ is specified by the following components. The set $\mathcal{X}$ is the set of possible *instances*, $\mathcal{Y}$ is the set of possible *labels*, $\mathcal{D}$ is a class of distributions containing some joint distributions $D \in \mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$.[10] The set $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ is called the *hypothesis space* or *hypothesis class*. We consider *loss functions* $\mathrm{Loss}\colon \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}_+$ where $\mathrm{Loss}(y', y)$ measures how different the 'prediction' y' (of some possible hypothesis $h(x) = y'$) is from the true outcome y.[11] We call a loss function *bounded* if it always takes values in $[0, 1]$. A natural loss function for classification tasks is to use $\mathrm{Loss}(y', y) = 0$ if $y = y'$ and $\mathrm{Loss}(y', y) = 1$ otherwise. For a given distribution $D \in \mathcal{D}$, the *risk* of a hypothesis $h \in \mathcal{H}$ is the expected loss of h with respect to D, namely $\mathrm{Risk}_D(h) = \mathbf{E}_{(x,y)\leftarrow D}[\mathrm{Loss}(h(x), y)]$.

An *example* s is a pair $s = (x, y)$ where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. An example is usually sampled from a distribution D. A *sample* set (or sequence) $\mathcal{S}$ of size n is a set (or sequence) of n examples. A hypothesis h is *consistent* with a sample set (or sequence) $\mathcal{S}$ if and only if $h(x) = y$ for all $(x, y) \in \mathcal{S}$. We assume that instances, labels, and hypotheses are encoded as strings over some alphabet such that given a hypothesis h and an instance x, $h(x)$ is computable in polynomial time.

Definition 1 (Realizability)**.** We say that the problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \mathrm{Loss})$ is realizable, if for all $D \in \mathcal{D}$, there exists an $h \in \mathcal{H}$ such that $\mathrm{Risk}_D(h) = 0$.

We can now define *Probably Approximately Correct (PAC)* learning. Our definition is with respect to a given set of distributions $\mathcal{D}$, and it can be instantiated with one distribution $\{D\} = \mathcal{D}$ to get the distribution-specific case. We can also recover the distribution-independent scenario, whenever the projection of $\mathcal{D}$ over $\mathcal{X}$ covers all distributions.

Definition 2 (PAC Learning)**.** A realizable problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \mathrm{Loss})$ is (ε, δ)-PAC learnable if there is a (possibly randomized) learning algorithm L such that for every n and every $D \in \mathcal{D}$, it holds that

$$\Pr_{\mathcal{S}\leftarrow D^n, h\leftarrow L(\mathcal{S})}[\mathrm{Risk}_D(h) \leq \varepsilon(n)] \geq 1 - \delta(n).$$

We call P simply PAC learnable if $\varepsilon(n), \delta(n) \leq 1/\mathrm{poly}(n)$, and we call it *efficiently* PAC learnable if, in addition, L is polynomial time.

Definition 3 (Average Error of a Test)**.** For a problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \mathrm{Loss})$, a (possibly randomized) learning algorithm L, a fixed test sample $(x, y) = d \leftarrow D$ for some distribution S over $\mathrm{Supp}(D)^n$ (e.g., $S \equiv D^n$) for some $n \in \mathbb{N}$, the *average error*[12] of the test example d (with respect to S, L) is defined as:

$$\mathrm{Err}_{S,L}(d) = \mathop{\mathbf{E}}_{\mathcal{S}\leftarrow S, h\leftarrow L(\mathcal{S})}[\mathrm{Loss}(h(x), y)].$$

When L is clear from the context, we simply write $\mathrm{Err}_S(d)$ to denote $\mathrm{Err}_{S,L}(d)$.

It is easy to see that a realizable problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \mathrm{Loss})$ with bounded Loss is PAC learnable iff there is a learner L (for P) such that the average of test's error $\gamma = \mathbf{E}_{d\leftarrow D}[\mathrm{Err}_{D^n}(d)]$ is bounded by a fixed $1/\mathrm{poly}(n)$ function for all $D \in \mathcal{D}$.[13]

Poisoning Attacks. PAC learning under adversarial noise is already defined in the literature, however, poisoning attacks include broader classes of attacks. For example, a poisoning adversary might *add* adversarial examples to the training data (thus, increasing it) or *remove* some of it adversarially. A more powerful form of poisoning attack is the so called *targeted* poisoning attacks where the adversary gets to know the targeted test example before poisoning the training examples. More formally, suppose $\mathcal{S} = (d_1, \ldots, d_n)$ is the training examples iid sampled from $D \in \mathcal{D}$. For a poisoning attacker A, by $\widehat{\mathcal{S}} \leftarrow \mathsf{A}(\mathcal{S})$ we denote the process

[10] By using joint distributions over $\mathcal{X} \times \mathcal{Y}$, we jointly model a set of distributions over $\mathcal{X}$ and a concept class mapping $\mathcal{X}$ to $\mathcal{Y}$ (perhaps with noise and uncertainty).

[11] Natural loss functions such as the 0-1 loss or the square loss assign the same amount of loss for same labels computed by h and c regardless of x.

[12] The work [27] called the same notion the 'cost' of d.

[13] Suppose $\mathrm{Loss}(\cdot)$ is bounded (i.e., always in $[0, 1]$). On one hand, if P is (ε, δ)-PAC learnable, then γ is at most $\varepsilon + \delta$. On the other hand, L is an $(\sqrt{\gamma}, \sqrt{\gamma})$-PAC learner.

through which A generates $\widehat{\mathcal{S}}$ based on $\mathcal{S}$. Note that, this notation does not specify the exact limitations of how A is allowed to tamper with $\mathcal{S}$, and that is part of the definition of A. In the targeted case, the adversary A is also given a test example $(x, y) = d \leftarrow D$. So, we would denote this by writing $\widehat{\mathcal{S}} \leftarrow \mathsf{A}(d, \mathcal{S})$ to emphasize that d is the test example given as input to A. We usually use $\mathcal{A}$ to denote a general adversary class. Note that a particular adversary $\mathsf{A} \in \mathcal{A}$ might try to poison a training set $\mathcal{S}$ *based* on the knowledge of a problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \mathrm{Loss})$. On the other hand, because sometimes we would like to limit adversary's power based on the specific distribution D (e.g. by always choosing tampered data to be in $\mathrm{Supp}(D)$). By $\mathcal{A}_D \subseteq \mathcal{A}$ we denote the adversary *class* for D.

Definition 4 (Learning under poisoning)**.** Suppose $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \mathrm{Loss})$ is a problem, $\mathcal{A} = \cup_{D \in \mathcal{D}} \mathcal{A}_D$ is an adversary class, and L is a (possibly randomized) learning algorithm for P.

- **PAC learning under poisoning.** If problem P is realizable, then L is an (ε, δ)-PAC learning for P under poisoning attacks of $\mathcal{A}$, if for every $D \in \mathcal{D}, n \in \mathbb{N}$, and every adversary $\mathsf{A} \in \mathcal{A}_D$:
$$\Pr_{\mathcal{S} \leftarrow D^n, \widehat{\mathcal{S}} \leftarrow \mathsf{A}(\mathcal{S}), h \leftarrow L(\widehat{\mathcal{S}})}[\mathrm{Risk}_D(h) \le \varepsilon(n)] \ge 1 - \delta(n).$$
PAC learnability and efficient PAC learnability are then defined similarly to Definition 2.
- **Average error under targeted poisoning.** If $\mathcal{A}$ contains *targeted* poisoning attackers, for a distribution $D \in \mathcal{D}$, and an attack $\mathsf{A} \in \mathcal{A}_D$ the *average error* $\mathrm{Err}^{\mathsf{A}}_{D^n}(d)$ for a test example $d = (x, y)$ under poisoning attacker A is $\mathrm{Err}_{\widehat{\mathcal{S}}}(d)$ where $\widehat{\mathcal{S}} \equiv \mathsf{A}(d, D^n)$.

We now define the class of poisoning attacks studied in this work. Informally speaking, p-tampering attacks model attackers who will manipulate the training sequence $\mathcal{S} = (d_1, \ldots, d_n)$ in an *online* way, meaning while tampering with d_i, they do not rely on the knowledge of $d_j, j > i$. Moreover, such attacks get to tamper with d_i only with independent probability p, modeling scenarios where the tampering even is random and outside the adversary's choice. A crucial point about p-tampering attacks is that they always stay in $\mathrm{Supp}(D)$. The formal definition follows.

Definition 5 (p-tampering/resetting attacks)**.** The class of p-tampering attacks $\mathcal{A}^p_{\mathrm{tam}} = \cup_{D \in \mathcal{D}} \mathcal{A}_D$ is defined as follows. For a distribution $D \in \mathcal{D}$, any $\mathsf{A} \in \mathcal{A}_D$ has a (potentially randomized) tampering algorithm Tam such that (1) given oracle access to D, $\mathsf{Tam}^D(\cdot) \in \mathrm{Supp}(D)$, and (2) given any training sequence $\mathcal{S} = (d_1, \ldots, d_n)$, the tampered $\widehat{\mathcal{S}} = (\widehat{d}_1, \ldots, \widehat{d}_n)$ is generated by A inductively (over $i \in [n]$) as follows:

- With probability $1 - p$, let $\widehat{d}_i = d_i$.
- Otherwise (this happens with probability p), output $\widehat{d}_i \leftarrow \mathsf{Tam}^D(1^n, \widehat{d}_1, \ldots, \widehat{d}_{i-1}, d_i)$.

The class of p-*resetting* attacks $\mathcal{A}^p_{\mathrm{res}} \subset \mathcal{A}^p_{\mathrm{tam}}$ include special cases of p-tampering attacks where the tampering algorithm Tam is restricted as follows. Either $\mathsf{Tam}(1^n, \widehat{d}_1, \ldots, \widehat{d}_{i-1}, d_i)$ outputs d_i, or otherwise, it will output a *fresh* sample $d'_i \leftarrow D$. In the *targeted* case, the adversary A_D and its tampering algorithm Tam are also given the final test example $d_0 \leftarrow D$ as extra input (that they can read but not tamper with). An attacker A_D is called *efficient*, if its oracle-aided tampering algorithm Tam^D runs in polynomial time.

Subtle aspects of the definition. Even though one can imagine a more general definition for tampering algorithms, in all the attacks of [27] and the attacks of this work, the tampering algorithms do *not* need to know the original un-tampered values $d_1, \ldots, d_{i-1}$. Since our goal here is to design p-tampering attacks, we use the simplified definition above. However, all of our positive results hold for the stronger variant in which the tampering algorithm is given the full history of the tampered and untampered blocks. Another subtle issue is about whether d_i is needed to be given to the tampering algorithm. As already noted in [27], when we care about p-tampering distributions of D^n, d_i does not necessarily have to be given to the tampering algorithm Tam, as Tam can itself sample a copy from D and treat it like d_i. Therefore the 'stronger' form of such attacks (where d_i is given) is equivalent to the 'weaker' form where d_i is not given. In fact, if D is efficiently samplable, then this equivalence holds with respect to efficient adversaries (with efficient Tam algorithm) as well. In this work, for both p-tampering and p-resetting attacks we choose to always give d_i to Tam. Interestingly, we show that if the adversary can *choose* the $p \cdot n$ locations of tampering, the weak and strong attackers will have different power!

Improved p-Tampering and p-Resetting Poisoning Attacks

In this section, we study the power of p-tampering attacks in the targeted setting and improve upon the p-tampering and p-resetting attacks of [27]. Our main tool is the following theorem giving new improved p-tampering and p-resetting attacks to bias the output of bounded real-valued functions.

Theorem 1 (Improved biasing attacks)**.** Let D be any distribution, $S \equiv D^n$, and $f\colon \mathrm{Supp}(S) \to [0, 1]$. Suppose $\mu = \mathbf{E}[f(S)]$ and $\nu = \mathrm{Var}[f(S)]$ be the average and the variance of $f(S)$ respectively. For every $p \in (0, 1)$, there is a p-tampering attack $\mathsf{A}_{\mathrm{tam}}$ such that
$$\mathop{\mathbf{E}}_{\widehat{S} \leftarrow \mathsf{A}_{\mathrm{tam}}(S)}[f(\widehat{S})] \ge \mu + \frac{p \cdot \nu}{1 + p \cdot \mu - p}$$
and a p-resetting attacker $\mathsf{A}_{\mathrm{res}}$ achieving bias of $\frac{p \cdot \nu}{1 + p \cdot \mu}$. Moreover, if D is efficiently sampleable and $f(\cdot)$ is efficiently computable, then $\mathsf{A}_{\mathrm{tam}}$ (resp. $\mathsf{A}_{\mathrm{res}}$) could be

implemented in time $\text{poly}(|D| \cdot n/\varepsilon)$ (where $|D|$ is the bit length of $d \leftarrow D$) while achieving bias at least $\frac{p\cdot\nu}{1+p\cdot\mu-p} - \varepsilon$ (resp. $\frac{p\cdot\nu}{1+p\cdot\mu} - \varepsilon$).

By using our improved biasing attacks, we can obtain the following improved attacks in the targeted setting against any learner. In particular, for any fixed $(x, y) = d \leftarrow D$, the following corollary follows from Theorem 1 by letting $f(\mathcal{S}) = \mathbf{E}_{h\leftarrow L(\mathcal{S})}[\text{Loss}(h(x), y)]$.

Corollary 2 (Improved targeted p-tampering attacks)**.** Given a problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \text{Loss})$ with a bounded loss function Loss, for any distribution $D \in \mathcal{D}$, test example $(x, y) = d \leftarrow D$, learner L, and $n \in \mathbb{N}$, let $\mu = \text{Err}_D(d)$ be the *average error* of d, and let

$$\nu = \underset{\mathcal{S}\leftarrow D^n}{\text{Var}}\left[\underset{h\leftarrow L(\mathcal{S})}{\mathbf{E}}[\text{Loss}(h(x), y)]\right].$$

Then, there is a p-tampering (resp. p-resetting) attack A_{tam} (resp. A_{res}) that increases the average error $\mu = \text{Err}_D(d)$ by $\frac{p\cdot\nu}{1+p\cdot\mu-p}$ (resp. $\frac{p\cdot\nu}{1+p\cdot\mu}$). Moreover, if D is efficiently samplable and f, Loss are efficiently computable, then $\mathsf{A}_{\text{tam}}, \mathsf{A}_{\text{res}}$ could achieve arbitrarily close biases in polynomial time.

Even if the average error $\mu = \text{Err}_D(d)$ is not small, the variance ν (as defined in Theorem 2) could be negligible. However, for natural cases this cannot happen. For example, if the loss function $\text{Loss}(\cdot)$ is Boolean (e.g., P is a classification) and if L is a deterministic learner, then $\nu = \mu \cdot (1 - \mu)$.

We also demonstrate the power of p-tampering and p-resetting attacks on PAC learners by using them to increase the failure probability of deterministic PAC learners. In particular, the following corollary follows from Theorem 1 by letting $f(\mathcal{S}) = 1$ if $\text{Risk}_D(h) \geq \varepsilon$ and $f(\mathcal{S}) = 0$ otherwise.

Corollary 3 (p-tampering attacks on PAC learners)**.** Given a problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \text{Loss})$, $D \in \mathcal{D}, n \in \mathbb{N}$, and deterministic learner L, suppose $\Pr_{\mathcal{S}\leftarrow D^n,\, h=L(\mathcal{S})}[\text{Risk}_D(h) \geq \varepsilon] = \delta$. Then, there a p-tampering attack A_{tam} such that

$$\underset{\widehat{\mathcal{S}}\leftarrow\mathsf{A}_{\text{tam}}(D^n), h=L(\widehat{\mathcal{S}})}{\Pr}[\text{Risk}_D(h) \geq \varepsilon] \geq \delta + \frac{p\cdot(\delta - \delta^2)}{1 + p\cdot\delta - p}$$

and a p-resetting A_{res} that can achieve bias of $\frac{p\cdot(\delta-\delta^2)}{1+p\cdot\delta}$. Moreover, if D is efficiently samplable and L, Loss are efficiently computable, then both of $\mathsf{A}_{\text{tam}}, \mathsf{A}_{\text{res}}$ could be implemented in polynomial time and make $\text{Risk}_D(h) \geq 0.99 \cdot \varepsilon$ happen with similar probabilities.

Our p-Tampering and p-Resetting Biasing Attacks: Proving Theorem 1

In this section, we prove Theorem 1. Recall Definition 5 and that the p-tampering attacker has an internal 'tampering' algorithm Tam that is executed with independent probability p. In this section, we only focus on describing the relevant tampering algorithms Tam and the general attacks will be defined accordingly. We will first describe our tampering algorithms in an ideal model where the certain parameters (see Definition 6) of the function f are given for free by an oracle. In the full version [26] we eliminate this idealized assumption by approximating the needed parameters efficiently.

Definition 6 (Average prefix function $\hat{f}$)**.** Let D be an distribution, $f\colon \text{Supp}(S) \mapsto \mathbb{R}$ be defined over D^n for some $n \in \mathbb{N}$, and $d_{\leq i} \in \text{Supp}(D)^i$ for some $i \in [n]$. We define the following functions.

- $f_{d_{\leq i}}(\cdot)$ is a function defined as $f_{d_{\leq i}}(d_{\geq i+1}) = f(z)$ where $z = (d_{\leq i}, d_{\geq i+1}) = (d_1, \ldots, d_n)$.
- $\hat{f}[d_{\leq i}] = \mathbf{E}_{d_{\geq i+1}\leftarrow D^{n-i}}[f_{d_{\leq i}}(d_{\geq i+1})]$. We also use $\mu = \hat{f}[\emptyset]$ to denote $\hat{f}[d_{\leq 0}] = \mathbf{E}[f(S)]$.

The key idea in both of our attacks is to design them (efficiently) based on oracle access to $\hat{f}$. The point is that $\hat{f}$ could later be approximated withing arbitrarily small $1/\text{poly}(n)$ factors, thus leading to sufficiently close approximations of our attacks. Due to lack of space, we will only sketch the main ideas that are used in making the attacks efficient in the appendix.

New p-Tampering Biasing Attack

In both of our attacks, we describe our attacks using functions with range $[-1, +1]$ instead. To get the results of Theorem 1 we simply need to scale the parameters back appropriately. Our Ideal p-Tam attack below, might repeat a loop indefinitely, but as we show in the full version [26], we can cut this rejection sampling procedure after a large enough polynomial iterations.

Construction 7 (Ideal p-Tam tampering)**.** Let D be an arbitrary distribution and $S \equiv D^n$ for some $n \in N$. Also let $f\colon \text{Supp}(D)^n \mapsto [-1, +1]$ be an arbitrary function. For any $i \in [n]$, given a prefix $d_{\leq i-1} \in \text{Supp}(D)^{i-1}$,[14] *ideal* p-Tam is a p-tampering attack defined as follows.

- Let $r[d_{\leq i}] = \frac{1-\hat{f}[d_{\leq i}]}{3-p-(1-p)\cdot\hat{f}[d_{\leq i-1}]}$.
- With probability $1 - r[d_{\leq i}]$ return d_i. Otherwise, sample a fresh $d_i \leftarrow D$ and go to step 1.

Proposition 4. Ideal p-Tam attack is well defined. Namely, $r[d_{\leq i}] \in [0, 1]$ for all $d_{\leq i} \in \text{Supp}(D)^i$.

Proof. Both $\hat{f}[d_{\leq i}], \hat{f}[d_{\leq i-1}]$ are in $[-1, 1]$. Therefore $0 \leq 1 - \hat{f}[d_{\leq i}] \leq 2$ and $3 - p - (1 - p) \cdot \hat{f}[d_{\leq i-1}] \geq 2$ which implies $0 \leq r[d_{\leq i}] \leq 1$. $\square$

[14]Note that here d_i is the 'original' untampered value for block i, while $d_1, \ldots, d_{i-1}$ might be the result of tampering.

In the following, let $\mathsf{A}_{\mathrm{tam}}$ be the p-tampering adversary using tampering algorithm.[15]

Claim 5. Let $\widehat{S} = (\widehat{D}_1, \ldots, \widehat{D}_n)$ be the joint distribution after $\mathsf{A}_{\mathrm{tam}}$ attack is performed on $S \equiv D^n$ using ideal p-Tam tampering algorithm. For all z, we have

$$\Pr[\widehat{S} = z] = \frac{2 - p + p \cdot f(z)}{2 - p + p \cdot \mu} \cdot \Pr[S = z].$$

Proof. During its execution, ideal p-Tam keeps sampling examples and rejecting them until a sample is accepted. For $\ell \in \mathbb{N}$ let R_ℓ be the event that the ℓ'th sample in the tampering algorithm is rejected, conditioned on reaching the ℓ th sample. Thus, $\Pr[\mathsf{R}_\ell]$ is equal to:

$$\begin{aligned}&\sum_{d_i} \Pr[D = d_i] \cdot \left(\frac{1 - \hat{f}[d_{\le i}]}{3 - p - (1-p) \cdot \hat{f}[d_{\le i-1}]} \right) \\ &= \frac{\sum_{d_i} \Pr[D = d_i] \cdot (1 - \hat{f}[d_{\le i}])}{3 - p - (1-p) \cdot \hat{f}[d_{\le i-1}]} \\ &= \frac{1 - \hat{f}[d_{\le i-1}]}{3 - p - (1-p) \cdot \hat{f}[d_{\le i-1}]}.\end{aligned}$$

Let $c[d_{\le i-1}] = \frac{1 - \hat{f}[d_{\le i-1}]}{3-p-(1-p) \cdot \hat{f}[d_{\le i-1}]}$ and $t[d_{\le i}] = \frac{\Pr[\widehat{D}_i = d_i | d_{\le i-1}]}{\Pr[D = d_i]}$. Then $t[d_{\le i}]$ is equal to:

$$\begin{aligned}&1 - p + p \cdot \left(\sum_{j=0}^{\infty} (1 - r[d_{\le i}]) \cdot \prod_{\ell=1}^{j} \Pr[\mathsf{R}_\ell] \right) \\ &= 1 - p + p \cdot \left(\sum_{j=0}^{\infty} (1 - r[d_{\le i}]) \cdot c[d_{\le i-1}]^j \right) \\ &= 1 - p + p \cdot \left(\frac{1 - r[d_{\le i}]}{1 - c[d_{\le i-1}]} \right) = \frac{2 - p + p \cdot \hat{f}[d_{\le i}]}{2 - p + p \cdot \hat{f}[d_{\le i-1}]}.\end{aligned}$$

Now, Claim 5 follows by a simple induction on i. □

Corollary 6. The p-tampering attack $\mathsf{A}_{\mathrm{tam}}$ (based on Ideal p-Tam tampering algorithm) biases $f(\cdot)$ by $\frac{p \cdot \nu}{2-p+p \cdot \mu}$ where $\mu = \mathbf{E}[f(S)], \nu = \mathrm{Var}[f(S)]$.

Proof. $\mathbf{E}[f(\widehat{S})] = \sum \Pr[\widehat{S} = z] \cdot f(z)$ is equal to

$$\begin{aligned}&\sum_{z \in \mathrm{Supp}(D)^n} \frac{2 - p + p \cdot f(z)}{2 - p + p \cdot \mu} \cdot \Pr[S = z] \cdot f(z) \\ &= \frac{2-p}{2 - p + p \cdot \mu} \cdot \sum_{z \in \mathrm{Supp}(D)^n} \Pr[S = z] \cdot f(z) \\ &\quad + \frac{p}{2 - p + p \cdot \mu} \cdot \sum_{z \in \mathrm{Supp}(D)^n} \Pr[S = z] \cdot f(z)^2.\end{aligned}$$

[15]Therefore, A_D, inductively runs p-Tam over the current sequence with probability p. See Definition 5.

This equal to:

$$\frac{(2-p) \cdot \mu}{2 - p + p \cdot \mu} + \frac{p \cdot (\nu + \mu^2)}{2 - p + p \cdot \mu} = \mu + \frac{p \cdot \nu}{2 - p + p \cdot \mu}.$$

□

New p-Resetting Biasing Attack

Construction 8 (Ideal p-Res)**.** Let D be an arbitrary distribution and $S \equiv D^n$ for some $n \in N$. Also let $f \colon \mathrm{Supp}(D)^n \mapsto [-1, +1]$. For any $i \in [n]$, and given a prefix $d_{\le i-1} \in \mathrm{Supp}(D)^{i-1}$, the p-Res tampering algorithm works as follows.

- Let $r[d_{\le i}] = \frac{1 - \hat{f}[d_{\le i}]}{2 + p \cdot (1 + \hat{f}[d_{\le i-1}])}$.
- With probability $1 - r[d_{\le i}]$ return the given d_i. Otherwise return a fresh sample $d'_i \leftarrow D$ (i.e., 'reset' d_i).

Proposition 7. Ideal p-Res algorithm is well defined. I.e., $r[d_{\le i}] \in [0, 1]$ for all $d_{\le i} \in \mathrm{Supp}(D)^i$.

Proof. We have $\hat{f}[d_{\le i}] \in [-1, +1]$ and $\hat{f}[d_{\le i-1}] \in [-1, +1]$. Therefore $0 \le 1 - \hat{f}[d_{\le i}] \le 2$ and $2 + p \cdot (1 + \hat{f}[d_{\le i-1}]) \ge 2$ which implies $0 \le r[d_{\le i}] \le 1$. □

In the following let $\mathsf{A}_{\mathrm{res}}$ be the p-tampering adversary using ideal p-Res. (See Definition 5.)

Claim 8. Let $\widehat{S} = (\widehat{D}_1, \ldots, \widehat{D}_n)$ be the distribution after the attack $\mathsf{A}_{\mathrm{res}}$ (using ideal p-Res tampering algorithm) is performed on $S \equiv D^n$. For all z, we have

$$\Pr[\widehat{S} = z] = \frac{2 + p + p \cdot f(z)}{2 + p + p \cdot \mu} \cdot \Pr[S = z].$$

Proof. We define R_0 to be the event that is true if the given sample is rejected. We have

$$\begin{aligned}\Pr[\mathsf{R}_0] &= \sum_{d_i} \Pr[D = d_i] \cdot \left(\frac{1 - \hat{f}[d_{\le i}]}{2 + p \cdot (1 + \hat{f}[d_{\le i-1}])} \right) \\ &= \frac{\sum_{d_i} \Pr[D = d_i] \cdot (1 - \hat{f}[d_{\le i}])}{2 + p \cdot (1 + \hat{f}[d_{\le i-1}])} \\ &= \frac{1 - \hat{f}[d_{\le i-1}]}{2 + p \cdot (1 + \hat{f}[d_{\le i-1}])}.\end{aligned}$$

Therefore, $\frac{\Pr[\widehat{D}_i = d_i | d_{\le i-1}]}{\Pr[D = d_i]}$ is equal to:

$$\begin{aligned}&= 1 - p + p \cdot (1 - r[d_{\le i}] + \Pr[\mathsf{R}_0]) \\ &= 1 - p + p \cdot \left(1 + \frac{\hat{f}[d_{\le i}] - \hat{f}[d_{\le i-1}]}{2 + p \cdot (1 + \hat{f}[d_{\le i-1}])} \right) \\ &= 1 + p \cdot \left(\frac{\hat{f}[d_{\le i}] - \hat{f}[d_{\le i-1}]}{2 + p \cdot (1 + \hat{f}[d_{\le i-1}])} \right) \\ &= \frac{2 + p \cdot (1 + \hat{f}[d_{\le i}])}{2 + p \cdot (1 + \hat{f}[d_{\le i-1}])}.\end{aligned}$$

Claim 8 then follows by a simple induction on i. □

Corollary 9. The p-resetting attack $\mathsf{A}_{\mathrm{res}}$ (using ideal p-Res tampering algorithm) biases the function by $\frac{p\cdot\nu}{2+p+p\cdot\mu}$ where $\mu = \mathbf{E}[f(S)], \nu = \mathrm{Var}[f(S)]$.

Proof. $\mathbf{E}[f(\widehat{S})] = \sum \Pr[\widehat{S} = z] \cdot f(z)$ is equal to:

$$\sum_{z\in\mathrm{Supp}(D)^n} \frac{2+p+p\cdot f(z)}{2+p+p\cdot\mu} \cdot \Pr[S=z]\cdot f(z)$$

$$= \frac{2+p}{2+p+p\cdot\mu} \cdot \sum_{z\in\mathrm{Supp}(D)^n} \Pr[S=z]\cdot f(z)$$

$$+ \frac{p}{2+p+p\cdot\mu} \cdot \sum_{z\in\mathrm{Supp}(D)^n} \Pr[S=z]\cdot f(z)^2$$

$$= \frac{(2+p)\cdot\mu}{2+p+p\cdot\mu} + \frac{p(\nu+\mu^2)}{2+p+p\cdot\mu} = \mu + \frac{p\cdot\nu}{2+p+p\cdot\mu}.$$

□

Feasibility of PAC Learning under Non-Targeted Poisoning Attacks

In this section, we study the non-targeted case where PAC learning could be defined. We show that realizable problems that are PAC learnable (without attacks), are usually PAC learnable under p-tampering attacks as well. Essentially we bound the probability of some bad event happening (see Definition 10) in a manner similar to Occam algorithms [10] by relying on the realizability assumption and relying on the specific property of the p-tampering attacks. In particular, we crucially rely on the fact that any p-tampering distribution $\widehat{D}$ of a distribution D contains a $(1-p)\cdot D$ measure in itself.

Definition 9. For problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \mathrm{Loss})$, distribution $D \in \mathcal{D}$, and training sequence $\mathcal{S} = ((x_1, y_1), \ldots, (x_n, y_n)) \leftarrow D^n$, we say that the event $\mathsf{Bad}_\varepsilon(D, \mathcal{S})$ holds, if there exists an $h \in \mathcal{H}$ such that $h(x_i) = y_i$ for every $i \in [n]$ and $\mathrm{Risk}_D(h) > \varepsilon$.

Definition 10 (Special PAC Learnability)**.** A realizable problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \mathrm{Loss})$ is called *special* $(\varepsilon(n), \delta(n))$-PAC learnable if for all $D \in \mathcal{D}, n \in \mathbb{N}$, $\Pr_{\mathcal{S}\leftarrow D^n}[\mathsf{Bad}_\varepsilon(D, \mathcal{S})] \leq \delta(n)$. Special $(\varepsilon(n), \delta(n))$-PAC learnability under poisoning attacks is defined similarly, where we demand the inequality to hold for every $\mathsf{A} \in \mathcal{A}_D$ tampering with the training set $\widehat{\mathcal{S}} \leftarrow \mathsf{A}(\mathcal{S})$.

It is easy to see that if P is special $(\varepsilon(n), \delta(n))$-PAC learnable, then it is $(\varepsilon(n), \delta(n))$-PAC learnable through a 'canonical' learner L who simply finds and outputs a hypothesis h consistent with the training sample set $\mathcal{S}$. Such an h always exists due to the realizability assumption. In fact, many *efficient* PAC learning results follow this very recipe.[16] That motivates our next definition.

[16]For example, properly learning monomials [39], or using 3-CNF formulae to learn 3-term DNF formulae [31]; the latter is an example of realizable but not proper learning. As an example where the realizability assumption does not necessarily hold, see e.g., [16], for learning monotone monomials under a class of distributions - including uniform.

Definition 11 (Efficient Realizability)**.** We say that the problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \mathrm{Loss})$ is *efficiently* realizable, if there is a polynomial-time algorithm M, such that for all $D \in \mathcal{D}$, and all $\mathcal{S} \leftarrow D^n$, $M(\mathcal{S})$ outputs some $h \in \mathcal{H}$ such that $\mathrm{Risk}_D(h) = 0$.

Theorem 10. For any $p \in (0, 1)$, if a realizable problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \mathrm{Loss})$ is $(\varepsilon(n), \delta(n))$-special PAC learnable, then for any $q \in (0, 1-p)$, P is also $(\varepsilon'(m), \delta'(m))$-special PAC learnable under p-tampering poisoning attacks for $\varepsilon'(m) = \varepsilon(m\cdot(1-p-q)), \delta'(m) = \mathrm{e}^{-2m\cdot q^2} + \delta(m\cdot(1-p-q))$. Thus, if P is efficiently realizable and special PAC learnable, then P is also efficiently PAC learnable under p-tampering.

Proof. Suppose we sample $\mathcal{S} \leftarrow D^m$. By a Chernoff bound, an adversary that tampers with each of the examples in $\mathcal{S}$ independently with probability p, will not change more than a $p+q$ fraction of the elements of $\mathcal{S}$ except with probability at most e^{-2mq^2}. Thus, with high probability, at least $(1-p-q)\cdot m \geq n$ examples in the tampered training sequence $\widehat{\mathcal{S}}$ are sampled from D *without* any control from the adversary. Since P is special $(\varepsilon(n), \delta(n))$-PAC learnable, with probability at least $1-\delta(n)$, these n 'untampered' examples from D will eliminate any hypothesis with risk larger than ε. Since the tampered sequence $\widehat{\mathcal{S}}$ of a p-tampering attack is in the support set $\mathrm{Supp}(D)^n$, due to realizability, there is at least one h such that $\mathrm{Risk}_D(h) = 0$. Hence, the learner can still find and output at least one $h \in \mathcal{H}$ for which $\mathrm{Risk}_D(h) \leq \varepsilon$. If further, P is efficiently realizable, h can be found in polynomial time. □

Bounded budget attackers. A p-tampering attacker does not have full control over which training examples become tamperable, as they each become tamperable with independent probability p. In the full version of this work [26] we further define two types of tampering attackers who *do* have control over which examples they tamper with, yet with a 'bounded budged' limiting the number of such instances. Our definitions are inspired by the notions of *adaptive corruption* [12,15,20,25] and *strong* adaptive corruption [20] originally defined in the context of secure multi-party coin-flipping protocols.

In particular, we show that in *strong* bounded budget p-tampering in which the adversary can choose the ($\leq p$ fraction of the) tampering locations, PAC learning might suddenly become impossible. This shows that the 'mistake-free' nature of p-tampering is indeed *not* enough for PAC learnability.[17]

[17]Bounded-budget noise has also been discussed outside of PAC learning; e.g., in the membership query model of [1].

References

[1] Dana Angluin. Queries and Concept Learning. *Machine Learning*, 2(4):319–342, 1987. 8

[2] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *Theory of cryptography*, pages 137–156, 2007. 2

[3] Per Austrin, Kai-Min Chung, Mohammad Mahmoody, Rafael Pass, and Karn Seth. On the impossibility of cryptography with tamperable randomness. In *International Cryptology Conference*, pages 462–479. Springer, 2014. 3

[4] Pranjal Awasthi, Maria Florina Balcan, and Philip M Long. The power of localization for efficiently learning linear separators with noise. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 449–458. ACM, 2014. 1, 2

[5] Salman Beigi, Omid Etesami, and Amin Gohari. Deterministic randomness extraction from generalized and distributed santha–vazirani sources. *SIAM Journal on Computing*, 46(1):1–36, 2017. 3

[6] Gyora M. Benedek and Alon Itai. Learnability with Respect to Fixed Distributions. *Theoretical Computer Science*, 86(2):377–390, 1991. 1

[7] Iddo Bentov, Ariel Gabizon, and David Zuckerman. Bitcoin beacon. *arXiv preprint arXiv:1605.04559*, 2016. 2, 3

[8] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering*, 26(4):984–996, 2014. 2

[9] Anselm Blumer, A. Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, October 1989. 1

[10] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's Razor. *Information Processing Letters*, 24(6):377–380, 1987. 8

[11] Nader H. Bshouty, Nadav Eiron, and Eyal Kushilevitz. PAC learning with nasty noise. *Theoretical Computer Science*, 288(2):255–275, 2002. 1

[12] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th Annual ACM Symposium on Theory of Computing*, pages 639–648, Philadephia, PA, USA, May 22–24, 1996. ACM Press. 3, 8

[13] Nicholas Carlini and David A. Wagner. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 39–57, 2017. 2

[14] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. In *Proc. 26th FOCS*, pages 429–442. IEEE, 1985. 3

[15] Richard Cleve and Russell Impagliazzo. Martingales, collective coin flipping and discrete control processes. *In other words*, 1:5, 1993. 8

[16] Dimitrios I. Diochnos. On the Evolution of Monotone Conjunctions: Drilling for Best Approximations. In *ALT*, pages 98–112, 2016. 8

[17] Dodis, Ong, Prabhakaran, and Sahai. On the (im)possibility of cryptography with imperfect randomness. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 2004. 3

[18] Yevgeniy Dodis and Yanqing Yao. Privacy with imperfect randomness. In *Annual Cryptology Conference*, pages 463–482. Springer, 2015. 3

[19] Andrzej Ehrenfeucht, David Haussler, Michael J. Kearns, and Leslie G. Valiant. A General Lower Bound on the Number of Examples Needed for Learning. *Information and Computation*, 82(3):247–261, 1989. 1

[20] Shafi Goldwasser, Yael Tauman Kalai, and Sunoo Park. Adaptively secure coin-flipping, revisited. In *International Colloquium on Automata, Languages, and Programming*, pages 663–674. Springer, 2015. 3, 8

[21] Carlos R. González and Yaser S. Abu-Mostafa. Mismatched training and test distributions can outperform matched ones. *Neural Computation*, 27(2):365–387, 2015. 2

[22] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *ICLR*, 2015. 2

[23] Iftach Haitner, Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions of protocols for secure computation. Cryptology ePrint Archive, Report 2010/164, 2010. `http://eprint.iacr.org/2010/164`. 2

[24] Michael J. Kearns and Ming Li. Learning in the Presence of Malicious Errors. *SIAM Journal on Computing*, 22(4):807–837, 1993. 1

[25] David Lichtenstein, Nathan Linial, and Michael Saks. Some extremal problems arising from discrete control processes. *Combinatorica*, 9(3):269–287, 1989. 8

[26] Saeed Mahloujifar, Dimitrios I. Diochnos, and Mohammad Mahmoody. Learning un-

der p-tampering attacks. *arXiv preprint arXiv:1711.03707*, 2017. 3, 6, 8

[27] Saeed Mahloujifar and Mohammad Mahmoody. Blockwise p-tampering attacks on cryptographic primitives, extractors, and learners. Theory of Cryptography Conference (TCC) 2017. Cryptology ePrint Archive, Report 2017/950, 2017. `http://eprint.iacr.org/2017/950`. 2, 3, 4, 5

[28] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *CVPR*, pages 2574–2582, 2016. 2

[29] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. 2

[30] Blaine Nelson, Benjamin I.P. Rubinstein, Ling Huang, Anthony D. Joseph, Steven J. Lee, Satish Rao, and J.D. Tygar. Query strategies for evading convex-inducing classifiers. *Journal of Machine Learning Research*, 13(May):1293–1332, 2012. 2

[31] Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988. 8

[32] Omer Reingold, Salil Vadhan, and Avi Wigderson. A note on extracting randomness from santha-vazirani sources. *Unpublished manuscript*, 2004. 3

[33] Benjamin I.P. Rubinstein, Blaine Nelson, Ling Huang, Anthony D. Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J.D. Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 1–14. ACM, 2009. 1

[34] Benjamin I.P. Rubinstein, Blaine Nelson, Ling Huang, Anthony D. Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J.D. Tygar. Stealthy poisoning attacks on pca-based anomaly detectors. *ACM SIGMETRICS Performance Evaluation Review*, 37(2):73–74, 2009. 1

[35] Miklos Santha and Umesh V. Vazirani. Generating quasi-random sequences from semi-random sources. *J. Comput. Syst. Sci.*, 33(1):75–87, 1986. 3

[36] Shiqi Shen, Shruti Tople, and Prateek Saxena. A uror: defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 508–519. ACM, 2016. 1, 2

[37] Robert H. Sloan. Four Types of Noise in Data for PAC Learning. *Information Processing Letters*, 54(3):157–162, 1995. 2

[38] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014. 2

[39] Leslie G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142, 1984. 1, 8

[40] Leslie G. Valiant. Learning disjunctions of conjunctions. In *IJCAI*, pages 560–566, 1985. 1

[41] John Von Neumann. 13. various techniques used in connection with random digits. *Appl. Math Ser*, 12:36–38, 1951. 3

[42] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *ICML*, pages 1689–1698, 2015. 1, 2

[43] Huan Xu and Shie Mannor. Robustness and generalization. *Machine Learning*, 86(3):391–423, 2012. 2

[44] Weilin Xu, David Evans, and Yanjun Qi. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. *CoRR*, abs/1704.01155, 2017. 2

[45] Keisuke Yamazaki, Motoaki Kawanabe, Sumio Watanabe, Masashi Sugiyama, and Klaus-Robert Müller. Asymptotic bayesian generalization error when training and test distributions are different. In *ICML*, pages 1079–1086, 2007. 2