Learning under p-Tampering Attacks

Saeed Mahloujifar[*] SAEED@VIRGINIA.EDU
Dimitrios I. Diochnos DIOCHNOS@VIRGINIA.EDU
Mohammad Mahmoody[†] MOHAMMAD@VIRGINIA.EDU
University of Virginia

Editor: Editor's name

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 only choose 'adversarial' examples 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 output of any bounded real-valued function through p-tampering.

In this work, we present new biasing attacks for biasing the average output of bounded real-valued functions. Our new biasing attacks achieve in *polynomial-time* the 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 attacks come with considerably 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 generated hypothesis (for a random test example). We show that PAC learning is *possible* under p-tampering poisoning attacks essentially whenever it is possible in the realizable setting without the attacks. We further show that PAC learning under 'no-mistake' adversarial noise is *not* possible, if the adversary could choose the (still limited to only p fraction of) tampered examples that she substitutes with adversarially chosen ones. Our formal model for such 'bounded-budget' tampering attackers is inspired by the notions of (strong) adaptive corruption in secure multi-party computation.

Keywords: Poisoning Attacks, Adversarial Learning, PAC Learning, Biasing Attacks

1. Introduction

In his seminal work (Valiant, 1984) Valiant introduced the Probably Approximately Correct (PAC) model of learning that triggered a significant amount of work around the theory of machine learning.[1] An important characteristic of learning algorithms is their ability to cope with noise. Valiant also initiated a study of adversarial noise (Valiant, 1985) in which each incoming training example is chosen, with independent probability p, by an adversary. Since

[*] Supported by University of Virginia's SEAS Research Innovation Award.

[†] Supported by NSF CAREER CCF-1350939, University of Virginia's SEAS Research Innovation Award.

[1] The original model studies learnability in a distribution-free sense, it also make sense for classes of distributions; (Benedek and Itai, 1991).

no assumptions are made on such modified examples, this type of noise is called *malicious*. Subsequently, (Kearns and Li, 1993) essentially proved impossibility of PAC learning under such malicious noise by heavily relying on the existence of *mistakes* (i.e., wrong labels) in adversarial examples given to the learner under a carefully chosen distribution. Bshouty, et al. (Bshouty et al., 2002) 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 (Kearns and Li, 1993) 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 still influence this distribution in (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 (Awasthi et al., 2014; Xiao et al., 2015; Shen et al., 2016) 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 (Rubinstein et al., 2009b,a) and the adversary has some noticeable chance of injecting or substituting malicious training data in an online manner. A stronger form of poisoning attacks are the so called *targeted* (poisoning) attacks (Shen et al., 2016), 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 (Awasthi et al., 2014; Xiao et al., 2015; Shen et al., 2016), the recent work of Mahloujifar and Mahmoody (Mahloujifar and Mahmoody, 2017) presented a generic *black-box* targeted poisoning attack that could adapt to apply to *any learner*, so long as there is an initial error over the target.

p-tampering attacks. The work of (Mahloujifar and Mahmoody, 2017) 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 (Sloan, 1995)) 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

[2] This is similar to (Blumer et al., 1989; Ehrenfeucht et al., 1989) that deals with the sample complexity.

[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.

[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.

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 (Haitner et al., 2010; Aumann and Lindell, 2007).

Poisoning through biasing. At the heart of the attacks of (Mahloujifar and Mahmoody, 2017) against learners was a basic p-tampering attack for *biasing* the average output of bounded real-valued functions. In particular, (Mahloujifar and Mahmoody, 2017) 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. (Note that the bias shall somehow depend on $\mathrm{Var}[f(S)]$ since constant functions cannot be biased.) 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, (Mahloujifar and Mahmoody, 2017) derived their p-tampering targeted poisoning attacks from them by 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.

Robustness. The robustness of a learner (Xu and Mannor, 2012; Yamazaki et al., 2007; González and Abu-Mostafa, 2015) refers to its behavior when the test examples are drawn from a distribution close to the training distribution but not necessary the same. 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.

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 (Szegedy et al., 2014). Such resulting misclassified perturbed instances are called *adversarial examples* and attacks aimed at finding such examples are called *evasion attacks* (Biggio et al., 2014; Nelson et al., 2012; Goodfellow et al., 2015; Moosavi-Dezfooli et al., 2016; Carlini and Wagner, 2017; Xu et al., 2017). The goal of evasion attacks is quite different from poisoning attacks: in poisoning attacks the tampering happens over the training data, while in evasion attacks no tempering to the training data is allowed but it is allowed for the test example itself. More work has also been done toward designing learning strategies that can achieve near optimal accuracy in presence of such attacks (Feige et al., 2015; Mansour et al., 2014).

1.1. Our Results

Improved p-tampering biasing attacks. Our main technical result in this work is to improve the polynomial-time p-tampering biasing attack of (Mahloujifar and Mahmoody, 2017) 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 6). 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 7). As in (Mahloujifar and Mahmoody, 2017), our attacks are black-box and apply to any learning problem P and any learner L for P as long as L has a non-negligible error over a specific test example d.

Special case of p-resetting attacks. The biasing attack of (Mahloujifar and Mahmoody, 2017) has an extra property that: 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 resampling 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 (Bentov et al., 2016) in the context of (ruling out) extracting uniform randomness from Bitcoin's blockchain (Nakamoto, 2008) when the adversary controls p fraction of the computing power, and thus it has the chance p of obtaining the next block, which she can discard/reset.[6] (Bentov et al., 2016) showed how to achieve bias $p/12$ when the original (untampered) distribution D is uniform and the function f is Boolean and balanced.[7] 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 got 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 their 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 (Mahloujifar and Mahmoody, 2017).

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.[8] 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 for 'realizable' settings; that is when there is always a hypothesis consistent with the training data (see Theorem 15).

We further 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 $\le p\cdot n$ 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 class of attacks strong/weak p-budget tampering attacks (see Definition 14). Our notions of p-budget tampering are inspired by notions of (strong) adaptive corruption (Canetti et al., 1996; Goldwasser et al., 2015) in cryptographic context. Our impossibility of PAC readability under strong p-budget attacks (see Theorem 16) shows that PAC learning under 'mistake-free' adversarial noise is *not*

[6] To compare the terminologies, the work of (Bentov et al., 2016) studies *p-resettable* sources of randomness, while here we study p-resetting attackers that generate such sources.

[7] The running time of the p-resetting attacker of (Bentov et al., 2016) was $\mathrm{poly}(n, 2^{|D|})$ where $|D|$ is the length of the binary representation of any $d \leftarrow D$, but our p-resetting attacks run in time $\mathrm{poly}(n, |D|)$.

[8] In the targeted setting, pre-selection of the target test eliminate the ε parameter of (ε,δ)-PAC learning.

always possible. Using our biasing attacks, we also obtain p-tampering and p-resetting attackers that increase the failure probability of any PAC learner (see Corollary 8).

Applications beyond attacking learners. Similar to how (Mahloujifar and Mahmoody, 2017) 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 (Von Neumann, 1951; Chor and Goldreich, 1985; Santha and Vazirani, 1986), as well as blockwise p-tampering (and p-resetting) attacks against security of certain cryptographic primitives (e.g., encryption, secure computation, etc.). As in (Mahloujifar and Mahmoody, 2017), 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. We refer the reader to the work of (Mahloujifar and Mahmoody, 2017) for such applications.

Ideas behind Our Biasing Attacks. The attacks of (Mahloujifar and Mahmoody, 2017), at a high level, were simple to describe, while their analysis were extremely complicated and heavily relied on carefully chosen potential functions based on ideas from (Austrin et al., 2014) 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 built upon ideas developed in previous work (Reingold et al., 2004; Dodis et al., 2004; Beigi et al., 2017; Dodis and Yao, 2015; Bentov et al., 2016) in the context of attacking deterministic randomness extractors from Santha-Vazirani sources (Santha and Vazirani, 1986). In (Mahloujifar and Mahmoody, 2017) the authors generalized the idea of 'half-space' sources (introduced in (Reingold et al., 2004; Dodis et al., 2004)) 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* is the main technical challenge resolved in this work.

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 (Mahloujifar and Mahmoody, 2017) 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 (Mahloujifar and Mahmoody, 2017) involves '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 $\hat{f}[d_{\leq i}]$ 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 (Mahloujifar and Mahmoody, 2017)). 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. See Sections A and Section A.3 for the details.

2. 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 $\mathcal{X} \times \mathcal{Y}$.[9] 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.[10] 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 $\mathrm{Risk}_D(h) = \mathbf{E}_{(x,y) \leftarrow D}\left[\mathrm{Loss}(h(x), y)\right]$.

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.

[9] 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).

[10] 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.

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[11] *of the test example* d *(with respect to* S, L*) is defined as:* $\mathrm{Err}_{S,L}(d) = \mathbf{E}_{\mathcal{S} \leftarrow S, h \leftarrow L(\mathcal{S})}[\mathrm{Loss}(h(x), y)]$*. 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 average error $\gamma = \mathbf{E}_{d \leftarrow D}[\mathrm{Err}_{D^n}(d)]$ is bounded by a fixed $1/\mathrm{poly}(n)$ function for all $D \in \mathcal{D}$.[12]

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 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 picking tampered data from $\mathrm{Supp}(D)$). By $\mathcal{A}_D \subseteq \mathcal{A}$ we denote the adversary *class* for D.

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

- **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* $\mathsf{A} \in \mathcal{A}_D$*:* $\Pr_{\mathcal{S} \leftarrow D^n, \widehat{\mathcal{S}} \leftarrow \mathsf{A}(\mathcal{S}), h \leftarrow L(\widehat{\mathcal{S}})}[\mathrm{Risk}_D(h) \leq \varepsilon(n)] \geq 1 - \delta(n)$*. 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 equal to* $\mathrm{Err}_{\widehat{S}}(d)$ *where* $\widehat{S} \equiv \mathsf{A}(d, S)$ *for* $S \equiv D^n$*.*

[11] The work (Mahloujifar and Mahmoody, 2017) called the same notion the 'cost' of d.

[12] 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.

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 $\mathcal{S} = (d_1, \dots, d_n)$ 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 $\operatorname{Supp}(D)$. 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, $\mathsf{Tam}^D(\cdot) \in \operatorname{Supp}(D)$, and (2) given any training sequence $\mathcal{S} = (d_1, \dots, d_n)$, the tampered $\widehat{\mathcal{S}} = (\widehat{d}_1, \dots, \widehat{d}_n)$ 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), get $\widehat{d}_i \leftarrow \mathsf{Tam}^D(1^n, \widehat{d}_1, \dots, \widehat{d}_{i-1}, d_i)$.*

*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 $\mathsf{Tam}(1^n, \widehat{d}_1, \dots, \widehat{d}_{i-1}, d_i)$ 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.*

Even though one can imagine a more general definition for tampering algorithms, in all the attacks of (Mahloujifar and Mahmoody, 2017) and the attacks of this work, the tampering algorithms do *not* need to know the original un-tampered values $d_1, \dots, d_{i-1}$. Since our goal here is to design p-tampering attacks, we use the simplified definition above, while all of our positive results for the stronger version in which the tampering algorithm is given the full history of the tampering algorithm. Another subtle issue is about whether d_i is needed to be given to the tampering algorithm. As already noted in (Mahloujifar and Mahmoody, 2017), when we care about p-tampering distributions of D^n, d_i is not necessary 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-resetting and p-resetting attacks we choose to always give d_i to Tam. Interestingly, as we show, if the adversary can *choose* the $p \cdot n$ locations of tampering, the weak and strong attackers will have different powers!

3. 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 (Mahloujifar and Mahmoody, 2017). 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 6 (Improved biasing attacks) *Let D be any distribution, $S \equiv D^n$, and $f \colon \operatorname{Supp}(S) \to [0, 1]$. Suppose $\mu = \mathbf{E}[f(S)]$ and $\nu = \operatorname{Var}[f(S)]$ be the average and the* *variance of $f(S)$ respectively. For every constant $p \in (0,1)$, there is a p-tampering attack* $\mathsf{A}_{\mathrm{tam}}$ *such that* $\mathbf{E}_{\widehat{\mathcal{S}} \leftarrow \mathsf{A}_{\mathrm{tam}}(S)}[f(\widehat{\mathcal{S}})] \geq \mu + \frac{p \cdot \nu}{1+p \cdot \mu - p}$ *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* $\mathrm{poly}(|D| \cdot n/\varepsilon)$ *(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.* $\frac{p \cdot \nu}{1+p \cdot \mu} - \varepsilon$*).*

We first describe the corollaries of the above theorem. We will then describe the actual attacks of Theorem 6. See Section A and Section A.3 for the full proof of Theorem 6.

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 6 by letting $f(\mathcal{S}) = \mathbf{E}_{h \leftarrow L(\mathcal{S})}[\mathrm{Loss}(h(x), y)]$.

Corollary 7 (Improved targeted p-tampering attacks) *Let* $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \mathrm{Loss})$ *be a problem 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* $\mu = \mathrm{Err}_D(d)$ *be the* average error *of d, and let* $\nu = \mathrm{Var}_{\mathcal{S} \leftarrow D^n}\left[\mathbf{E}_{h \leftarrow L(\mathcal{S})}[\mathrm{Loss}(h(x), y)]\right]$*. Then, there is a p-tampering (resp. p-resetting) attack* $\mathsf{A}_{\mathrm{tam}}$ *(resp.* $\mathsf{A}_{\mathrm{res}}$*) that increases the average error* $\mu = \mathrm{Err}_D(d)$ *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* $\mathsf{A}_{\mathrm{tam}}, \mathsf{A}_{\mathrm{res}}$ *could achieve arbitrarily close biases in polynomial time.*

Even if the average error $\mu = \mathrm{Err}_D(d)$ is not small, the variance ν (see Corollary 7) could be negligible. However, for some natural cases this cannot happen, e.g., if the loss function $\mathrm{Loss}(\cdot)$ is Boolean and L is deterministic, then $\nu = \mu \cdot (1 - \mu)$.

We also demonstrate the power of p-tampering and p-resetting attacks on PAC learners by increasing the error of deterministic PAC learners. In particular, the following corollary follows from Theorem 6 by letting $f(\mathcal{S}) = 1$ if $\mathrm{Risk}_D(h) \geq \varepsilon$ and $f(\mathcal{S}) = 0$ otherwise.

Corollary 8 (p-tampering attacks on PAC learners) *For* $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \mathrm{Loss})$*, let $D \in \mathcal{D}, n \in \mathbb{N}$, L be a deterministic learner for* P*, and suppose* $\Pr_{\mathcal{S} \leftarrow D^n,\ h = L(\mathcal{S})}[\mathrm{Risk}_D(h) \geq \varepsilon] = \delta$*. Then, there a p-tampering attack* $\mathsf{A}_{\mathrm{tam}}$ *and a p-resetting attack* $\mathsf{A}_{\mathrm{res}}$ *such that*

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

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

New biasing attacks. We now describe the high level structure of the attacks of Theorem 6. Recall Definition 5 and that the p-tampering attacker has an internal 'tampering' algorithm Tam that is executed with independent probability p. Thus, we only need to describe 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 9) of the function f are given for free by an oracle. In Section A.3 we eliminate this idealized assumption by approximating the oracle efficiently.

Definition 9 (Function $\hat{f}$) *Let* $f \colon \mathrm{Supp}(D^n) \mapsto \mathbb{R}$ *for distribution D and some $n \in \mathbb{N}$, and let* $d_{\leq i} \in \mathrm{Supp}(D)^i$ *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 $1/\operatorname{poly}(n)$ factors, thus leading to sufficiently close approximations of our attacks. After describing the 'ideal' version of the attacks, we will then make them efficient by approximating $\hat{f}$.

We describe both of the attacks using functions with range $[-1, +1]$ instead. To get the results of Theorem 6 we simply need to scale the parameters back appropriately.

Our Ideal p-Tam attack below, might repeat a loop indefinitely, in Section A.3, we show that one can cut this loop after a large enough polynomial number of rounds.

Construction 1 (Ideal p-Tam):
Let D be an arbitrary distribution, and let $f\colon \operatorname{Supp}(D)^n \mapsto [-1, +1]$ be an arbitrary function. For any $i \in [n]$, given a prefix $d_{\leq i-1} \in \operatorname{Supp}(D)^{i-1}$,[13] *ideal* p-Tam is a p-tampering attack defined as follows.

1. Let $r[d_{\leq i}] = \frac{1-\hat{f}[d_{\leq i}]}{3-p-(1-p)\cdot\hat{f}[d_{\leq i-1}]}$.
2. Return d_i with probability $1 - r[d_{\leq i}]$, otherwise sample $d_i \leftarrow D$ and go to step 1.

We now describe our p-resetting attack.

Construction 2 (Ideal p-Res):
Let D be an arbitrary distribution, and let $f\colon \operatorname{Supp}(D)^n \mapsto [-1, +1]$. For any $i \in [n]$, and given a prefix $d_{\leq i-1} \in \operatorname{Supp}(D)^{i-1}$, the p-Res tampering algorithm works as follows.

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

See Section A for full proof of Theorem 6 using attacks of Constructions 1 and 2.

4. Feasibility of PAC Learning under (Variants of) p-Tampering 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 11) in a manner similar to Occam algorithms (Blumer et al., 1987) 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. In fact, we show (see Theorem 16) that in a close scenario to 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.[14]

[13] 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.

[14] We note that bounded-budget noise and in fact malicious has also been discussed outside of PAC learning; e.g., (Angluin et al., 1997) in the membership query model of Angluin (Angluin, 1987).

Definition 10 *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 11 (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 the inequality holds 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.[15] That motivates our next definition.

Definition 12 (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 13 (PAC learning under p-tampering) *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 $\mathrm{Supp}(D)^n$, 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 a control over which training examples become tamperable, and they each become so with independent probability p. Here we 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* (Canetti et al., 1996)

[15] For example, properly learning monomials (Valiant, 1984), or using 3-CNF formulae to learn 3-term DNF formulae (Pitt and Valiant, 1988); 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., (Diochnos, 2016), for learning monotone monomials under a class of distributions - including uniform.

and *strong* adaptive corruption defined by Goldwasser, Kalai, and Park (Goldwasser et al., 2015) in the secure multi-party (coin-flipping) protocols.

Definition 14 (p-budget tampering) *The class of* strong *p-budget tampering attacks $\mathcal{A}^p_{\text{bud}} = \cup_{D \in \mathcal{D}} \mathcal{A}_D$ is defined as follows. For $D \in \mathcal{D}$, any $\mathsf{A} \in \mathcal{A}_D$ has a (randomized) tampering algorithm* Tam *such that:*

1. *Given oracle access to D, $\mathsf{Tam}^D(\cdot)$ always outputs something in $\mathrm{Supp}(D)$.*
2. *Given any training sequence $\mathcal{S} = (d_1, \dots, d_n)$, the tampered output $\widehat{\mathcal{S}} = (\widehat{d}_1, \dots, \widehat{d}_n)$ is generated by A inductively (over $i \in [n]$) as $\widehat{d}_i \leftarrow \mathsf{Tam}^D(1^n, \widehat{d}_1, \dots, \widehat{d}_{i-1}, d_i)$.*
3. *The number of location i where Tam changes d_i is bounded: $|\{i \mid d_i \neq \widehat{d}_i\}| \leq p \cdot n$.*

Weak p-budged tampering attacks are defined similarly, with the following difference. The tampering circuit's execution $\mathsf{Tam}^D(1^n, \widehat{d}_1, \dots, \widehat{d}_{i-1}, d_1, \dots, d_{i-1})$ is not *given d_i, but it could either output $\widehat{d}_i \in \mathrm{Supp}(D)$ or a special symbol $\perp$, in which case A_D will choose $\widehat{d}_i = d_i$.*

In Theorem 15, we extend Theorem 13 and prove that PAC learning is possible under weak p-budget poisoning attacks. This positive result holds even if the tampering algorithm is given all the history of tampered and untampered blocks (i.e., it is given given input $(1^n, \widehat{d}_1, \dots, \widehat{d}_{i-1}, d_1, \dots, d_i)$). See Section B for a proof.

Theorem 15 (PAC learning under weak p-budget attacks) *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, P is also $(\varepsilon(n \cdot (1-p)), \delta(n \cdot (1-p)))$-special PAC learnable under* weak *p-budget tampering.*

In contrast to Theorem 15, the following theorem shows that, in general, PAC learning (of PAC learnable problems) is *not* possible under *strong* p-budget poisoning attacks.

Theorem 16 (Impossibility of PAC learning under strong p-budget tampering) *For any constant $p \in (0,1)$, there is a problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{D}, \mathcal{H}, \mathrm{Loss})$ that is PAC learnable (when no attack happens), but it is* not *PAC learnable under strong p-budget tampering.*

Proof Suppose $\mathcal{X} = [k]$ for a constant k where $1/p < k \leq 2/p$. (Such integer k exists because $p < 1$ implies $2/p - 1/p > 1$.) Let $\mathcal{Y} = \{0,1\}$, and suppose $\mathcal{D}$ consists of all $(x, c(x))_{x \leftarrow \mathcal{X}}$ where $x \leftarrow \mathcal{X}$ is the uniform sample from $\mathcal{X}$ and c is an an arbitrary function (concept) in $\mathcal{Y}^{\mathcal{X}}$, $\mathcal{H}$ contains all of $\mathcal{Y}^{\mathcal{X}}$, and $\mathrm{Loss}(b_0, b_1) = |b_0 - b_1|$ is natural for classifiers.

PAC learnability of P trivially follows from the fact that $|\mathcal{X}| = k$ is finite. Therefore, enough samples will reveal the concept function c (defined through D) completely. On the other hand, consider two concepts c_0, c_1 where $c_0(x) = 0$ for all $x \in [k]$, and $c_1(x) = 0$ for all $x \in [k-1]$ and $c_1(k) = 1$. Let $D_0 \equiv (U, c_0(U))$. Consider the following strong p-budget tampering attacks A_D for $D \in \{D_0, D_1\}$: whenever $d_i = (k, b)$ for $b \in \{0,1\}$, A_D substitutes d_i with $\widehat{d}_i = (0,0)$. If A_D manages to tamper with all $d_i = (k, b)$ examples, then the (tampered) training examples would be identically distributed for both cases of D_0, D_1. On the other hand, the probability that A_D runs out of its $p \cdot n$ tampering budget is $2^{\Omega(-n)}$ which is at most $o(n)$ for sufficiently larger n. Therefore, if there is any $(\varepsilon(n), \delta(n))$ PAC learning for P under such strong p-tampering attacks, it would require $\varepsilon(n) + \delta(n) \geq \Omega(1/k) \geq \Omega(1/p)$. ■

References

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

Dana Angluin, Martins Krikis, Robert H. Sloan, and György Turán. Malicious Omissions and Errors in Answers to Membership Queries. *Machine Learning*, 28(2-3):211–255, 1997.

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

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.

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.

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.

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

Iddo Bentov, Ariel Gabizon, and David Zuckerman. Bitcoin beacon. *arXiv preprint arXiv:1605.04559*, 2016.

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.

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

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.

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

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.

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.

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.

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

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

Yevgeniy Dodis, Shien Jin Ong, Manoj Prabhakaran, and Amit Sahai. On the (Im)possibility of Cryptography with Imperfect Randomness. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 2004.

Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 51–60. IEEE, 2010.

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.

Uriel Feige, Yishay Mansour, and Robert Schapire. Learning and inference in the presence of corrupted inputs. In *Conference on Learning Theory*, pages 637–657, 2015.

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.

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

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *ICLR*, 2015. URL http://arxiv.org/abs/1412.6572.

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.

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

Saeed Mahloujifar and Mohammad Mahmoody. Blockwise p-tampering attacks on cryptographic primitives, extractors, and learners. In *Theory of Cryptography Conference*, pages 245–279. Springer, 2017.

Yishay Mansour, Aviad Rubinstein, and Moshe Tennenholtz. Robust probabilistic inference. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 449–460. SIAM, 2014.

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.

Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

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

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

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

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, 2009a.

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, 2009b.

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

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.

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

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014. URL http://arxiv.org/abs/1312.6199.

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

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

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

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.

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

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

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.

Appendix A. Full Proof of Theorem 6

In this section we prove Theorem 6.

A.1. Proving Theorem 6 for the p-Tampering Case

Here we prove that Construction 1 does have the properties stated for the p-tampering attack of Theorem 6 when we have access to the idealized oracle. In Section A.3 we remove this assumption by approximating the idealized oracle efficiently. All the notation below is with respect to Construction 1.

Proposition 17 *Ideal p-Tam attack is well defined. Namely, $r[d_{\leq i}] \in [0,1]$ for all $d_{\leq i} \in \mathrm{Supp}(D)^i$.*

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 $0 \leq r[d_{\leq i}] \leq 1$. ■

In the following, let $\mathsf{A}_{\mathrm{tam}}$ be the p-tampering adversary using tampering algorithm Ideal p-Tam.[16]

Claim 1:
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 every prefix $d_{\leq i} \in \mathrm{Supp}(D)^i$ we have:

$$\frac{\Pr[\widehat{D}_i = d_i \mid d_{\leq i-1}]}{\Pr[D = d_i]} = \frac{2 - p \cdot (1 - \hat{f}[d_{\leq i}])}{2 - p \cdot (1 - \hat{f}[d_{\leq i-1}])}.$$

Proof During its execution, ideal p-Tam keeps sampling examples and rejecting them until a sample is accepted. For $\ell \in \mathbb{N}$ we define R_ℓ to be the event that is true if the ℓ'th sample in the tampering algorithm is rejected, conditioned on reaching the ℓth sample. We have

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

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

Let $c[d_{\le i-1}] = \frac{1-\hat{f}[d_{\le i-1}]}{3-p-(1-p)\cdot\hat{f}[d_{\le i-1}]}$. Then we have

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

■

The following corollary follows from Claim 1 and induction.

Corollary 18 *By applying the attack $\mathsf{A}_{\mathrm{tam}}$ based on ideal p-Tam tampering algorithm, the distribution after the attack would be as follows*

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

Corollary 19 *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 It holds that $\mathbf{E}[f(\widehat{S})]$ is equal to:

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

■

A.2. Proving Theorem 6 for the p-Resetting Case

Here we prove that Construction 2 does have the properties stated for the p-resetting attack of Theorem 6 when we have access to the idealized oracle. In Section A.3 we remove this assumption by approximating the idealized oracle efficiently. All the notation below is with respect to Construction 2.

Proposition 20 *Ideal p-Res algorithm is well defined. I.e., $r[d_{\leq i}] \in [0,1]$ for all $d_{\leq i} \in \mathrm{Supp}(D)^i$.*

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

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

Claim 2:
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 $d_{\leq i} \in \mathrm{Supp}(D)^i$ it holds that:

$$\frac{\Pr[\widehat{D}_i = d_i \mid d_{\leq i-1}]}{\Pr[D = d_i]} = \frac{2 + p \cdot (1 + \hat{f}[d_{\leq i}])}{2 + p \cdot (1 + \hat{f}[d_{\leq i-1}])}.$$

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

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

Therefore, we conclude that:

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

■

The following corollary follows from Claim 2 and induction.

Corollary 21 *By applying attack $\mathsf{A}_{\mathrm{res}}$ (using ideal p-Res), the distribution after the attack is:*

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

Corollary 22 *The p-resetting attack $\mathsf{A}_{\mathrm{res}}$ (using ideal p-Res) 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 It holds that $\widehat{\mu} = \mathbf{E}[f(\widehat{S})]$ is equal to:

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

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

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

■

A.3. Approximating the Ideal Attacks in Polynomial Time

In this subsection, we describe the efficient version of the attacks of Theorem 6 and prove their properties. We first describe the efficient version of our p-resetting attack, where achieving efficiency is indeed simpler. We then go over the efficient variant of our p-tampering attack. In both cases, we describe the modifications needed for the *tampering algorithms* and it is assumed that such tampering algorithms are used by the main efficient attackers (see Definition 5).

A.3.1. Efficient p-Resetting Biasing

The p-resetting attack of Construction 2 is not efficient since it needs oracle access to the idealized oracle providing partial averages. In general, we can not compute such averages exactly in polynomial time, however in order to make those attacks efficient, we can rely on *approximating* the partial averages and consequently the corresponding rejection probabilities. To get the efficient version of the attack of Construction 2 we can pursue the following idea. For every prefix $d_{\leq i}$, the efficient attacker first approximates the partial average $\hat{f}[d_{\leq i}]$ by sampling a sufficiently large polynomial number of random continuations $d^{(1)}_{\leq n-i}, \dots d^{(\ell)}_{\leq n-i}$ and getting the average $\mathbf{E}_{j \in [\ell]}[f(d_{\leq i}, d^{(j)}_{\leq n-i}]$ as an approximation for the partial average. By Hoeffding inequality, this average is a good approximation of $\hat{f}[d_{\leq i}]$ with exponentially high probability. Consequently, the rejection probabilities can be approximated well making the final distributions statically close to the distribution of the ideal attack, meaning that the amount of bias is close to the ideal bias as well.

Now we formalize the ideas above.

Definition 23 (Semi-ideal oracle $\tilde{f}[\cdot]$) *For distribution D, if for all $d_{\leq i} \in \mathrm{Supp}(D)^i$ we have $\tilde{f}_\varepsilon[d_{\leq i}] \in \hat{f}[d_{\leq i}] \pm \varepsilon$, then, we call $\tilde{f}_\varepsilon[\cdot]$ an ε-approximation of $\hat{f}[\cdot]$. For simplicity, and when it is clear from the context, we simply write $\tilde{f}[\cdot]$ and call it a* semi-ideal *oracle.*

The following lemma immediately follows from the Hoeffding inequality.

Lemma 24 (Approximating $\hat{f}[\cdot]$ efficiently) *Consider an algorithm that on inputs $d_{\leq i}$ and ε performs as follows where $\ell = -10\ln(\varepsilon/2)/\varepsilon^2$.*

1. *Sample* $(d^1_{\leq n-i}, \dots, d^\ell_{\leq n-i}) \leftarrow (D^{n-i+1})^\ell$.

2. *Output* $\tilde{f}_\varepsilon[d_{\leq i}] = \mathbf{E}_{j \in [\ell]}\, f(d_{\leq i}, d^j_{\leq n-i})$.

Then it holds that $\Pr[|\tilde{f}_\varepsilon[d_{\leq i}] - \hat{f}[d_{\leq i}]| \geq \varepsilon] \leq \varepsilon$.

The above lemma implies that if f is efficiently computable and D is efficiently samplable, any q-query algorithm can approximate the semi-ideal oracle $\tilde{f}[\cdot]$ in time $\operatorname{poly}(q \cdot n/\varepsilon)$ and total error (of failing in one of the queries) by at most ε. Based on this efficient approximation of $\tilde{f}[\cdot]$, we now describe our efficient version of the Ideal p-Res attack in the semi-ideal oracle model of $\tilde{f}[\cdot]$, by essentially using the semi-ideal oracle $\tilde{f}[\cdot]$ instead of the ideal oracle $\hat{f}[\cdot]$.

Construction 3 (Efficient p-Res):
Efficient p-Res is the same as ideal p-Res of Construction 2 but it calls the semi-ideal oracle $\tilde{f}_\varepsilon[\cdot]$ instead of the ideal oracle $\hat{f}[\cdot]$.

In the following we analyze the bias achieved by the Efficient p-Res algorithm. We simply pretend that all the queries to the semi-ideal oracle are within $\pm\varepsilon$ approximation of the ideal oracle, knowing that the error of ε-approximating all of the queries is itself at most ε and can affect the average also by at most $O(\varepsilon)$. First we show that the rejection probabilities are approximated well.

Lemma 25 *Let $r[.]$ and $\tilde{r}[.]$ respectively be the rejection probabilities of the Ideal and Efficient p-Res. Then, for every $d_{\leq i} \in \operatorname{Supp}(D)^i$ we have $|r[d_{\leq i}] - \tilde{r}[d_{\leq i}]| \leq O(\varepsilon)$.*

Proof Let $p' \in p \pm \varepsilon, q' \in q \pm \varepsilon$ for $p, q \in (0,1)$. We first show that $\frac{p'}{1+q'} \in \frac{p}{1+q} \pm O(\varepsilon)$.

$$\left|\frac{p'}{1+q'} - \frac{p}{1+q}\right| = \left|\frac{p' - p + p' \cdot q - p \cdot q'}{(1+q) \cdot (1+q')}\right| \leq \left|p' - p + p' \cdot (q - q') + q' \cdot (p' - p)\right| \leq 3 \cdot \varepsilon$$

Now using this general statement we conclude that $|r[d_{\leq i}] - \tilde{r}[d_{\leq i}]| \leq 3 \cdot \varepsilon$. ■

Now we want to argue that when we approximate the p-resetting tampering algorithm's rejection probabilities as proved in Lemma 25, it leads to 'close probabilities' of sampling final outputs. We prove the following general lemma that will be also useful for the case of Efficient p-Tam attack. For the case of p-resetting, we only need the special case of $k = 1$.

Notation. For $p \in [0,1]$ and distributions X, Y, by $Z \equiv (1-p)X + pY$ we denote the distribution Z in which we sample from X with probability $1-p$, and otherwise (i.e., with probability p) we sample from Y.

Definition 26 ((p,k,ρ)-variations) *For any distribution D, function $\rho\colon \operatorname{Supp}(D) \to [0,1]$, and $k \in \mathbb{N}$, the (p,k,ρ)-variation of D is $D_{p,k,\rho} \equiv (1-p)D + pZ$, where Z is defined as follows.*

1. *Sample* $(d_1, \dots, d_k) \leftarrow D^k$.

2. *Sequentially go over $d_1, \dots, d_k$, and with probability $\rho[d_i]$ return d_i and exit.*

3. *If nothing was returned after reading all the k samples, return a fresh sample $d_{k+1} \leftarrow D$.*

Lemma 27 (Implication of approximating rejection probabilities) *Let D be a distribution and $\rho : \mathrm{Supp}(D) \to [0,1]$ and $\rho' : \mathrm{Supp}(D) \to [0,1]$ be two functions such that $\forall d \in \mathrm{Supp}(D), |\rho(d) - \rho'(d)| \le \varepsilon$. Then, for every $k \in \mathbb{N}$ and every $d \in \mathrm{Supp}(D)$, it holds that*

$$\left|\ln\left(\frac{\Pr[D_{p,k,\rho} = d]}{\Pr[D_{p,k,\rho'} = d]}\right)\right| \le \frac{p}{1-p} \cdot (k^2 + k) \cdot \varepsilon.$$

Before proving the lemma above, we note that it indeed implies that the *max divergence* (Dwork et al., 2010) of $D_{p,k,\rho}$ and $D_{p,k,\rho'}$ is at most $O(k^2 \cdot \varepsilon)$.

Proof Let $a = \mathbf{E}_{d \leftarrow D}[\rho(d)]$ and $a' = \mathbf{E}_{d \leftarrow D}[\rho'(d)]$. We have

$$\frac{\Pr[D_{p,k,\rho} = d]}{\Pr[D = d]} = (1-p) + p \cdot ((1-a)^k + \sum_{i \in [k-1]} \rho(d) \cdot (1-a)^i).$$

With a similar calculation for $\Pr[D_{p,k,\rho'} = d]$ we get

$$\begin{aligned}
\frac{\Pr[D_{p,k,\rho} = d]}{\Pr[D_{p,k,\rho'} = d]} &= \frac{(1-p) + p \cdot ((1-a)^k + \sum_{i \in [k-1]} \rho(d) \cdot (1-a)^i)}{(1-p) + p \cdot ((1-a')^k + \sum_{i \in [k-1]} \rho(d) \cdot (1-a')^i)} \\
&= 1 + \frac{p \cdot ((1-a)^k - (1-a')^k + \sum_{i \in [k-1]} \rho(d) \cdot (1-a)^i - \rho'(d) \cdot (1-a')^i)}{(1-p) + p \cdot ((1-a')^k + \sum_{i \in [k-1]} \rho(d) \cdot (1-a')^i)} \\
&\le 1 + \frac{p \cdot (k \cdot \varepsilon + \sum_{i \in [k-1]} (2i+1) \cdot \varepsilon)}{1-p} \\
&= 1 + \frac{p}{1-p}(k^2 + k) \cdot \varepsilon \\
&\le \mathrm{e}^{\frac{p}{1-p}(k^2+k)\cdot\varepsilon}.
\end{aligned}$$

Similarly, we have $\frac{\Pr[D_{p,k,\rho'}=d]}{\Pr[D_{p,k,\rho}=d]} \le \mathrm{e}^{\frac{p}{1-p}(k^2+k)\varepsilon}$ which implies that

$$\left|\ln\left(\frac{\Pr[D_{p,k,\rho} = d]}{\Pr[D_{p,k,\rho'} = d]}\right)\right| \le \frac{p}{1-p} \cdot (k^2 + k) \cdot \varepsilon.$$

■

The following lemma states that the averages of a function over two distributions that are 'close' (under max divergence) are indeed close real numbers.

Lemma 28 *Let $X = (X_1, \dots, X_n)$ and $Y = (Y_1, \dots, Y_n)$ be two joint distributions such that $\mathrm{Supp}(X) = \mathrm{Supp}(Y)$ and for every prefix $x_{\le i}$ such that $\Pr[X_i = x_i | x_{\le i-1}] > 0$, we have*

$$\left|\ln\left(\frac{\Pr[X_i = x_i \mid x_{\le i-1}]}{\Pr[Y_i = x_i \mid x_{\le i-1}]}\right)\right| \le \varepsilon.$$

Then, for any function $f\colon \mathrm{Supp}(X) \to [-1,+1]$ *we have*

$$\mathbf{E}[f(X)] \geq \mathbf{E}[f(Y)] - \mathrm{e}^{\varepsilon\cdot n} + 1.$$

Proof First, we note that for every $x \in \mathrm{Supp}(X)$ it holds that

$$\left|\ln\left(\frac{\Pr[X=x]}{\Pr[Y=x]}\right)\right| = \left|\sum_{i\in[n]} \ln\left(\frac{\Pr[X_i = x_i \mid x_{\leq i-1}]}{\Pr[Y_i = x_i \mid x_{\leq i-1}]}\right)\right| \leq n\cdot\varepsilon.$$

Now we can show that $\mathbf{E}[f(Y)] - \mathbf{E}[f(X)]$ is equal to

$$\begin{aligned}
&\sum_{x\in\mathrm{Supp}(X)} (\Pr[Y=x] - \Pr[X=x])\cdot f(x) \\
&\leq \sum_{x\in\mathrm{Supp}(X)} |(\Pr[Y=x] - \Pr[X=x])\cdot f(x)| \\
&\leq \sum_{x\in\mathrm{Supp}(X)} \left|\min(\Pr[X=x],\Pr[Y=x])\cdot\left(\frac{\max(\Pr[X=x],\Pr[Y=x])}{\min(\Pr[X=x],\Pr[Y=x])} - 1\right)\cdot f(x)\right| \\
&\leq (\mathrm{e}^{n\cdot\varepsilon} - 1)\cdot \sum_{x\in\mathrm{Supp}(X)} |\min(\Pr[X=x],\Pr[Y=x])\cdot f(x)| \leq e^{n\cdot\varepsilon} - 1.
\end{aligned}$$

■

Putting things together. Now we show how to choose the parameters of the Efficient p-Res. Suppose ε' is the parameter of Theorem 6. If we choose ε as the parameter of our attack we can bound the final bias as follows. Firstly, if the approximation algorithm of Lemma 24 gives us a semi-ideal oracle $\tilde{f}_\varepsilon[.]$, then based on Lemma 25 we can approximate the rejection probabilities with error at most $O(\varepsilon)$. Then based on Lemma 27 the attack $\mathsf{A}_{\mathrm{res}}$ that uses efficient p-Res generates a distribution that is $O(\frac{p}{1-p}\cdot\varepsilon)$-close to the distribution of the attack $\mathsf{A}_{\mathrm{res}}$ that uses ideal p-Res. Now we can use Lemma 28 (for $k=1$) to argue that bias of efficient adversary is $(\mathrm{e}^{O(n\cdot\varepsilon\cdot\frac{p}{1-p})} - 1)$-close to bias of ideal adversary. Also note that, if the approximation algorithm fails to provide a semi-ideal oracle for all queries, then bias of efficient attack is at least -2 because the function range is $[-1,+1]$. However, the probability of this event is bounded by $O(n\cdot\varepsilon)$ because adversary needs at most $2n$ number of queries to $\tilde{f}$. Therefore, the difference of bias of efficient and ideal adversary is at most $O(n\cdot\varepsilon) + \mathrm{e}^{O(n\cdot\varepsilon\cdot\frac{p}{1-p})} - 1$ which is at most $O(n\cdot\varepsilon + n\cdot\varepsilon\cdot\frac{p}{1-p})$ if the exponent in $\mathrm{e}^{O(n\cdot\varepsilon\cdot\frac{p}{1-p})}$ is at most 1. As a result, if we choose $\varepsilon = o(\varepsilon'/(n\cdot\frac{p}{1-p})) = o(\varepsilon'\cdot(1-p)/(n\cdot p))$, we can indeed guarantee that bias of efficient adversary is ε'-close to bias of ideal adversary.

A.3.2. EFFICIENT p-TAMPERING BIASING

Building upon the ideas developed above to make our Ideal p-Res tampering algorithm polynomial time, here we focus on our Ideal p-Tam attack. We start by describing a variant of the original attack of Construction 1 where we cut the rejection sampling procedure after k iterations.

Construction 4 (Ideal k-cut p-Tam):
Ideal k-cut p-Tam is the same as ideal p-Tam of Construction 1 but it is forced to stop and return a fresh sample if the first k samples were rejected.

Now we show that the new modified attack of Construction 4 will lead to a close distribution compared to the original attack of Construction 1.

Lemma 29 *Let $\widehat{S} = (\widehat{D}_1, \ldots, \widehat{D}_n)$ be the joint distribution after A_{tam} attack is performed on $S \equiv D^n$ using ideal p-Tam tampering algorithm. Also, let $\widehat{S}' = (\widehat{D}'_1, \ldots, \widehat{D}'_n)$ be the joint distribution after A_{tam} attack is performed on S using Ideal k-cut p-Tam tampering algorithm. For every prefix $d_{\leq i} \in \operatorname{Supp}(D)^i$:*

$$\left| \ln \left(\frac{\Pr[\widehat{D}_i = d_i \mid d_{\leq i-1}]}{\Pr[\widehat{D}'_i = d_i | d_{\leq i-1}]} \right) \right| \leq \frac{p}{(1-p)^2 \cdot (2-p)^{k-1}}.$$

Proof Let $r[d_{\leq i}] = \frac{1-\hat{f}[d_{\leq i}]}{3-p-(1-p)\cdot \hat{f}[d_{\leq i-1}]}$ and $c[d_{\leq i-1}] = \frac{1-\hat{f}[d_{\leq i-1}]}{3-p-(1-p)\cdot \hat{f}[d_{\leq i-1}]}$ as it was defined in proof of Claim 1. We have

$$\begin{aligned} \frac{\Pr[\widehat{D}'_i = d_i \mid d_{\leq i-1}]}{\Pr[D = d_i]} &= (1-p) + p \cdot \left((c[d_{\leq i-1}])^k + \sum_{j \in [k-1]} (1 - r[d_{\leq i}]) \cdot (1 - c[d_{\leq i})]^j) \right) \\ &= (1-p) + p \cdot \left((c[d_{\leq i-1}])^k + \frac{(1 - r[d_{\leq i}]) \cdot (1 - c[d_{\leq i-1}]^k)}{1 - c[d_{\leq i-1}]} \right). \end{aligned}$$

Also, in the proof of Claim 1 we showed that

$$\frac{\Pr[\widehat{D}_i = d_i \mid d_{\leq i-1}]}{\Pr[D = d_i]} = 1 - p + p \cdot \left(\frac{1 - r[d_{\leq i}]}{1 - c[d_{\leq i-1}]} \right).$$

Therefore, we conclude that

$$\begin{aligned} \frac{\Pr[\widehat{D}'_i = d_i \mid d_{\leq i-1}]}{\Pr[\widehat{D}_i = d_i \mid d_{\leq i-1}]} &= \frac{(1-p) + p \cdot \left((c[d_{\leq i-1}])^k + \frac{(1-r[d_{\leq i}])\cdot(1-c[d_{\leq i-1}]^k)}{1-c[d_{\leq i-1}]} \right)}{1 - p + p \cdot \left(\frac{1-r[d_{\leq i}]}{1-c[d_{\leq i-1}]} \right)} \\ &= 1 + \frac{p \cdot \left(\frac{(r[d_{\leq i}] - c[d_{\leq i-1}]) \cdot c[d_{\leq i-1}]^k}{1-c[d_{\leq i-1}]} \right)}{1 - p + p \cdot \left(\frac{1-r[d_{\leq i}]}{1-c[d_{\leq i-1}]} \right)}. \end{aligned}$$

We also know that $c[d_{\leq i-1}] \leq \frac{1}{2-p}$ because $\hat{f}[d_{\leq i-1}] \in [-1, +1]$. So we have

$$\begin{aligned} \frac{\Pr[\widehat{D}'_i = d_i \mid d_{\leq i-1}]}{\Pr[\widehat{D}_i = d_i \mid d_{\leq i-1}]} &= 1 + \frac{p \cdot \left(\frac{(r[d_{\leq i}] - c[d_{\leq i-1}]) \cdot c[d_{\leq i-1}]^k}{1-c[d_{\leq i-1}]} \right)}{1 - p + p \cdot \left(\frac{1-r[d_{\leq i}]}{1-c[d_{\leq i-1}]} \right)} \\ &\leq 1 + \frac{p \cdot c[d_{\leq i-1}]^k}{(1-p) \cdot (1 - c[d_{\leq i-1}])} \\ &\leq 1 + \frac{p}{(1-p)^2 (2-p)^{k-1}} \leq \mathrm{e}^{\frac{p}{(1-p)^2(2-p)^{k-1}}}. \end{aligned}$$

Also for the inverse ratio, we have

$$\begin{aligned}\frac{\Pr[\widehat{D}_i = d_i \mid d_{\leq i-1}]}{\Pr[\widehat{D}'_i = d_i \mid d_{\leq i-1}]} &= 1 + \frac{p \cdot \left(\frac{(c[d_{\leq i-1}] - r[d_{\leq i}]) \cdot c[d_{\leq i-1}]^k}{1 - c[d_{\leq i-1}]}\right)}{(1-p) + p \cdot \left((c[d_{\leq i-1}])^k + \frac{(1 - r[d_{\leq i}]) \cdot (1 - c[d_{\leq i-1}]^k)}{1 - c[d_{\leq i-1}]}\right)} \\ &\leq 1 + \frac{p \cdot c[d_{\leq i-1}]^k}{(1-p) \cdot (1 - c[d_{\leq i-1}])} \\ &\leq 1 + \frac{p}{(1-p)^2 (2-p)^{k-1}} \leq \mathrm{e}^{\frac{p}{(1-p)^2(2-p)^{k-1}}}.\end{aligned}$$

Therefore, we can finally conclude that

$$\left| \ln \left(\frac{\Pr[\widehat{D}_i = d_i \mid d_{\leq i-1}]}{\Pr[\widehat{D}'_i = d_i | d_{\leq i-1}]} \right) \right| \leq \frac{p}{(1-p)^2 (2-p)^{k-1}}.$$

■

Lemma 30 *Let $\widehat{S} = (\widehat{D}_1, \ldots, \widehat{D}_n)$ be the joint distribution after A_{tam} attack is performed on $S \equiv D^n$ using ideal p-Tam tampering algorithm. Also, let $\widehat{S}' = (\widehat{D}'_1, \ldots, \widehat{D}'_n)$ be the joint distribution after A_{tam} attack is performed on S using Ideal k-cut p-Tam tampering algorithm where $k = \frac{\ln(2-p) - 2\ln((1-p)\varepsilon)}{\ln(2-p)}$. Then:*

$$\mathbf{E}[f(\widehat{S}')] \geq \mathbf{E}[f(\widehat{S})] - \mathrm{e}^{n \cdot \varepsilon} + 1.$$

Proof Using Lemma 29, for every prefix $d_{\leq i} \in \mathrm{Supp}(D)^i$ we have:

$$\left| \ln \left(\frac{\Pr[\widehat{D}_i = d_i \mid d_{\leq i-1}]}{\Pr[\widehat{D}'_i = d_i | d_{\leq i-1}]} \right) \right| \leq \frac{p}{(1-p)^2 (2-p)^{k-1}} \leq \varepsilon.$$

Now, using Lemma 28 we get $\mathbf{E}[f(\widehat{S}')] \geq \mathbf{E}[f(\widehat{S})] - \mathrm{e}^{n \cdot \varepsilon} + 1$. ■

We can now describe the actual efficient variant of our Ideal p-Tam attack.

Construction 5 (Efficient k-cut p-Tam):
Efficient k-cut p-Tam is the same as Ideal k-cut p-Tam of Construction 4 but it it calls the semi-ideal oracle $\tilde{f}_\varepsilon[\cdot]$ instead of the ideal oracle $\hat{f}[\cdot]$.

Lemma 31 *Let $r[.]$ and $\tilde{r}[.]$ respectively be the rejection probabilities of the Ideal and Efficient k-cut p-Tam. Then, for every $d_{\leq i} \in \mathrm{Supp}(D)^i$ we have $|r[d_{\leq i}] - \tilde{r}[d_{\leq i}]| \leq O(\varepsilon)$.*

The proof of above Lemma is similar to the proof of Lemma 25.

Putting things together. Now we show how to choose the parameters of the Efficient k-cut p-Tam. Suppose ε' is the parameter of Theorem 6. If we choose ε as the parameter of our attack we can bound the final bias as follows. Firstly, if the approximation algorithm of Lemma 24 gives us a semi-ideal oracle $\tilde{f}_\varepsilon[.]$, then based on Lemma 31 we can approximate the rejection probabilities with error at most $O(\varepsilon)$. Then based on Lemma 27 the attack $\mathsf{A}_{\mathrm{tam}}$ that uses efficient k-cut p-Tam generates a distribution that is $O(\frac{p}{1-p} \cdot k^2 \cdot \varepsilon)$-close to the distribution of the attack $\mathsf{A}_{\mathrm{tam}}$ that uses ideal k-cut p-Tam. Now we can use Lemma 28 to argue that bias of efficient adversary is $\big(\mathrm{e}^{O(n\cdot\varepsilon\cdot k^2\cdot\frac{p}{1-p})} - 1\big)$-close to bias of ideal adversary. Also note that, if the approximation algorithm fails to provide a semi-ideal oracle for all queries, then bias of efficient attack is at least -2 because the function range is $[-1,+1]$. However, the probability of this event is bounded by $O(k \cdot n \cdot \varepsilon)$ because adversary needs at most $(k+1)\cdot n$ number of queries to $\tilde{f}$. Therefore, the difference of bias of efficient and ideal adversary is at most $O(k\cdot n\cdot\varepsilon) + \mathrm{e}^{O(k^2\cdot n\cdot\varepsilon\cdot\frac{p}{1-p})} - 1$ which is at most $O(n\cdot\varepsilon + k^2\cdot n\cdot\varepsilon\cdot\frac{p}{1-p})$ if the exponent in $\mathrm{e}^{O(k^2\cdot n\cdot\varepsilon\cdot\frac{p}{1-p})}$ is at most 1. As a result, if we choose $\varepsilon = o(\varepsilon'/(k^2\cdot n\cdot\frac{p}{1-p})) = o(\varepsilon'\cdot(1-p)/(k^2\cdot n\cdot p))$, we can indeed guarantee that bias of efficient adversary (that uses efficient k-cut p-Tam tampering algorithm) is ε'-close to bias of ideal adversary (that uses ideal k-cut p-Tam). Now we want to select our other parameter k. Based on Lemma 30, if we choose $k = \omega(\frac{\ln((1-p)\varepsilon')}{\ln(2-p)})$ the bias of attack $\mathsf{A}_{\mathrm{tam}}$ that uses ideal k-cut p-Tam would be ε'-close to the bias of attack $\mathsf{A}_{\mathrm{tam}}$ that uses ideal p-Tam. Therefore, the bias of the $\mathsf{A}_{\mathrm{tam}}$ that uses efficient k-cut attack is $2\cdot\varepsilon'$-close to the bias of $\mathsf{A}_{\mathrm{tam}}$ that uses ideal p-Tam.

Appendix B. Proof of Theorem 15

Proof Intuitively, in any weak p-budget tampering attack, the adversary can choose the locations of the tampering but has no control over the sampled d_i when d_i is not tampered with.[17] More formally, compare the actual attack to the following 'ideal' experiment. In the ideal experiment, we first choose $m = n\cdot(1-p)$ samples $e_1,\dots,e_m \leftarrow D$ before even running the adversary. Then, we run the adversary A_D who internally runs the tampering algorithm Tam inductively as follows. We let a counter initially $\ell = 0$ counting how many of e_i's we have used so far. For every $i \in [n]$, if $\mathsf{Tam}^D(1^n, \widehat{d}_1,\dots,\widehat{d}_{i-1}, d_1,\dots,d_{i-1})$ outputs $\perp$, then we increase ℓ by one choose the next e_ℓ, and if $\ell > m$ we simply use a fresh $d_i \leftarrow D$. It is easy to see that this ideal execution is statistically identical to the real attack experiment. On the other hand, because A_D has is p-budget, Tam will output $\perp$ at least $n\cdot(1-p)$ times, meaning that we will use all of e_i's (i.e. ℓ gets eventually increased to m). Because all the initial samples $e_1,\dots,e_m \leftarrow D$ eventually find their way into the tampered training example sequence $\widehat{\mathcal{S}}$, by the special PAC learnability of P, it holds that the same learner L for P still outputs with probability $1-\delta(m)$ a hypothesis with risk at most $\varepsilon(m)$. ■

[17] This seems to be also the case in strong p-budget attacks, but as we see in Theorem 16, the fact that the adversary can first see d_i and then choose not to tamper with them in the strong case, will allow her to essentially choose 'untampered' d_i's and prevent PAC learnability.