Multi-party Poisoning through Generalized p-Tampering

Saeed Mahloujifar* Mohammad Mahmoody† Ameer Mohammed‡

September 11, 2018

Abstract

In a poisoning attack against a learning algorithm, an adversary tampers with a fraction of the training data $\mathcal{T}$ with the goal of increasing the classification error of the constructed hypothesis/model over the final test distribution. In the distributed setting, $\mathcal{T}$ might be gathered gradually from m data providers $P_1, \ldots, P_m$ who generate and submit their shares of $\mathcal{T}$ in an online way.

In this work, we initiate a formal study of (k, p)-poisoning attacks in which an adversary controls $k \in [n]$ of the parties, and even for each corrupted party P_i, the adversary submits some poisoned data $\mathcal{T}_i'$ on behalf of P_i that is still "$(1-p)$-close" to the correct data $\mathcal{T}_i$ (e.g., $1-p$ fraction of $\mathcal{T}_i'$ is still honestly generated). For $k = m$, this model becomes the traditional notion of poisoning, and for $p = 1$ it coincides with the standard notion of corruption in multi-party computation.

We prove that if there is an initial constant error ε for the generated hypothesis h, there is always a (k, p)-poisoning attacker who can decrease the confidence of h (to have a small error), or alternatively increase the error of h, by $\Omega(p \cdot k/m)$. Our attacks can be implemented in polynomial time given samples from the correct data, and they use no wrong labels if the original distributions are not noisy.

At a technical level, we prove a general lemma about biasing bounded functions $f(x_1, \ldots, x_n) \in [0, 1]$ through an attack model in which each block x_i might be controlled by an adversary with marginal probability p in an online way. When the probabilities are independent, this coincides with the model of p-tampering attacks [ACM+17, MM17, MDM18], thus we call our model generalized p-tampering. We prove the power of such attacks by incorporating ideas from the context of coin-flipping attacks from [BOL89, HO14] into the p-tampering model and generalize the results in both of these areas.

*University of Virginia.
†University of Virginia.
‡Kuwait University.

Contents

1 Introduction **2**
- 1.1 Ideas Behind Our Generalized p-Tampering Attack 6

2 Multi-party Poisoning Attacks: Definitions and Main Results **7**
- 2.1 Basic Definitions for Multi-party Learning and Poisoning 8
- 2.2 Power of Multy-party Poisoning Attacks 9

3 Generalized p-Tampering Biasing Attacks: Definitions and Main Results **10**
- 3.1 Preliminary Notation and Basic Definitions for Tampering with Random Processes 10
- 3.2 Power of Generalized p-Tampering Attacks 11

4 Proofs of the Main Results **12**
- 4.1 Obtaining (k, p)-Poisoning Attacks: Proof of Theorem 2.4 12
- 4.2 Computationally Unbounded Attacks: Proving Part 1 of Theorem 3.6 13
- 4.3 Polynomial-time Attacks: Proving Part 2 of Theorem 3.6 15
- 4.4 Relating the Bias to the Variance: Proving Lemma 3.7 18

A Some Useful Inequalities **19**

1 Introduction

Learning from a set $\mathcal{T} = \{d_1, \ldots, d_n\}$ of training examples in a way that the predictions generalize to instances beyond $\mathcal{T}$ is a fundamental problem in learning theory. The goal here is to produce a hypothesis (also called a model) h in such a way that $h(a)$, with high probability, predicts the "correct" label b, where the pair $(a, b) = d$ is sampled from the target (test) distribution $\mathbf{d}$. In the most natural setting, the examples in the training data set $\mathcal{T}$ are also generated from the same distribution $\mathbf{d}$, however this is not always the case. For example, the examples in $\mathcal{T}$ could be gathered under noisy conditions. Even more, the difference between the distribution producing $\mathcal{T}$ and the test distribution $\mathbf{d}$ could be *adversarial*. Indeed, in his seminal work Valiant [Val85] initiated a formal study of this phenomenon by defining the framework of learning under *malicious noise*, that is a model in which an all-powerful adversary is allowed to change each of the generated training examples $d \in \mathcal{T}$ with independent probability p, in an online way. Subsequently, it was shown [KL93, BEK02] that PAC learning of even simple problems in this model could be impossible, at least for specific pathological distributions.

Poisoning attacks. A more modern interpretation for the problem of learning under adversarial noise is the framework of so-called *poisoning* (aka *causative*) attacks (e.g., see [ABL14, XBB+15, STS16, PMSW16]), in which the adversary's goal is not necessarily to completely prevent the learning, but perhaps it simply wants to increase the risk of the hypothesis produced by the learning process. A poisoning attack could be defined also in settings that are not at all covered by the malicious noise model; for example a poisoning attacker might even have a particular test example in mind while doing the whole attack, making the final attack a *targeted* one [STS16]. Mahloujifar, Mahmoody and Diochnos [MM17, MDM18] initiated a study of poisoning attacks in a model that closely follows Valiant's malicious noise model and showed that such attacks can indeed increase the error of any classifiers for any learning problem by a constant probability, *even without* using wrong labels, so long as there is an initial constant error probability. The attack model

used in [MM17,MDM18], called p-tampering, was a generalization of a similar model introduced in Austrin et al. [ACM+14] in the bitwise setting in cryptographic contexts.

Multi-party poisoning. In a distributed learning procedure [MR17, MMR+16, BIK+17, KMY+16], the training data $\mathcal{T}$ might be coming from various sources; e.g., it can be generated by m data providers $P_1, \ldots, P_m$ in an online way, while at the end a fixed algorithm, called the aggregator G, generates the hypothesis h based on $\mathcal{T}$. The goal of $P_1, \ldots, P_m$ is to eventually help G construct a hypothesis h that does well in predicting the label b of a given instance a, where $(a, b) \leftarrow \mathbf{d}$ is sampled from the final test distribution. The data provided by each party P_i might even be of "different type", so we cannot simply assume that the data provided by P_i is necessarily sampled from the same distribution $\mathbf{d}$. Rather, we let $\mathbf{d}_i$ model the distribution from which the training data $\mathcal{T}_i$ (of P_i) is sampled. Poisoning attacks can naturally be defined in the distributed setting as well (e.g., see [FYB18, BVH+18, BGS+17]) to model adversaries who partially control the training data $\mathcal{T}$ with the goal of decreasing the quality of the generated hypothesis. The central question of our work is then as follows.

> *What is the inherent power of poisoning attacks in the multi-party setting? How much can they increase the risk of the final trained hypothesis, if they only have a "limited" power?*

Using multi-party coin-tossing attacks? The question above could be studied in various settings, but the most natural way to model it from a cryptographic perspective is to allow the adversary to control k out of the m parties. So, a natural idea here is to use techniques from attacks in the context of multi-party coin-tossing protocols [BOL89, HO14]. Indeed, the adversary in that context wants to change the outcome of a random bit generated by m parties by corrupting k of them. So, at a high level, if we interpret the output bit $b = 1$ to be the case that h makes a mistake on its test and interpret $b = 0$ to be the other case, we might be able to use such attacks to increase the risk of h by increasing the probability of $b = 1$. At a high level, the issues with using this idea are that: (1) the attacks in the multi-party coin tossing are not polynomial time, while we need polynomial-time attacks, (2) they only apply to Boolean output, while here we might want to increase the loss function of h that is potentially real-valued, and finally (3) coin-tossing attacks, like other cryptographic attacks in the multi-party setting, completely change the messages of the corrupted parties, while here we might want to keep the corrupted distributions "close" to the original ones, perhaps with the goal of not alarming a suspicious behavior, or simply because the attack only gets *partial* control over the process by which P_i generates its data. Indeed, we would like to model such milder forms of attacks as well.

Using p-tampering attacks? Now, let us try a different approach assuming that the adversary's corrupted parties have a randomized pattern. In particular, let us assume that the adversary gets to corrupt and control k *randomly* selected parties. In this case, it is easy to see that, at the end every single message in the protocol Π between the parties $P_1, \ldots, P_m$ is controlled with exactly probability $p = k/m$ by the adversary Adv (even though these probabilities are correlated). Thus, at a high level it seems that we should be able to use the p-tampering attacks of [MM17, MDM18] to degrade the quality of the produced hypothesis. However, the catch is that the proof of p-tampering attacks of [MM17, MDM18] (and the bitwise version of [ACM+17]) crucially rely on the assumption that each message (which in our context corresponds to a training example) is tamperable with *independent* probability p, while by corrupting k random parties, the set of messages controlled by the adversary are highly correlated.

A new attack model: (k,p)-poisoning attacks. To get the best of p-tampering attacks and the coin-tossing attacks with k corrupted parties, we combine these models and define a new model, called (k,p)-poisoning, that generalizes the corruption pattern in both of these settings. A (k,p)-poisoning attacker Adv can first choose to corrupt k of the parties, but then even after doing so, Adv has a limited control over the training examples generated by a corrupted party. More formally, if a corrupted $\widetilde{P}_i$ is supposed to send the next message, then the adversary will sample $d \leftarrow \widetilde{\mathbf{d}}$ for a maliciously chosen distribution $\widetilde{\mathbf{d}}$ that is guaranteed to be "close" to the original distribution $\mathbf{d}_i$, while their distance is controlled by a parameter $p \in [0,1]$. In particular, we require that the statistical distance between $\widetilde{\mathbf{d}}$ and $\mathbf{d}_i$ is at most p. It is easy to see that (k,p)-poisoning attacks include p-tampering attacks where $k = m$ (m is the number of parties). Moreover, (k,p)-attacks trivially include k-corrupting attacks by letting $p = 1$. Our main result in this works is to prove the following general theorem about the *inherent* power of (k,p)-poisoning attacks.

Theorem 1.1 (Power of (k,p)-poisoning attacks–**informal**). *Let $\Pi = (P_1, \ldots, P_m)$ be an m-party learning protocol for an m-party learning problem. There is a polynomial time (k,p)-poisoning attack Adv such that, given oracle access to the data distribution of the parties, Adv can decrease the confidence of the learning process by $\Omega(p \cdot \frac{k}{m})$, where the confidence parameter is*

$$1 - \Pr[\text{the risk of the generated hypothesis is} > \alpha]$$

for a fixed parameter α.[1] Alternatively, for any target example d, there is a similar polynomial time (k,p)-poisoning attack that can increase the average error of final hypothesis on d by $\Omega(p \cdot \frac{k}{m})$, where this average is also over the generated hypothesis h.

(For the formal version of Theorem 1.1 above, see Theorem 2.4.)

We prove the above theorem by first proving a general result about the power of "biasing" adversaries whose goal is to increase the expected value of a random process by controlling each block of the random process with probability q (think of q as $\approx p \cdot k/m$). As these biasing attacks generalize p-tampering attacks, we simply call them *generalized* p-tampering attacks. We now describe this attack model and clarify how it can be used to achieve our goals stated in Theorem 1.1.

Generalized p-tampering biasing attacks. Generalized p-tampering attacks could be defined for any random process $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and a function $f(\overline{\mathbf{x}}) \in [0,1]$ defined over this process. In order to explain the attack model, first consider the setting where there is no attacker. Now, given a prefix $x_1, \ldots, x_{i-1}$ of the blocks, the next block x_i is simply sampled from its conditional probability distribution $(\mathbf{x}_i \mid x_1, \ldots, x_{i-1})$. (Looking ahead, think of x_i as the i'th training example provided by one of the parties in the interactive learning protocol.) Now, imagine an adversary who enters the game and whose goal is to increase the expected value of a function $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ defined over the random process $\overline{\mathbf{x}}$ by tampering with the block-by-block sampling process of $\overline{\mathbf{x}}$ described above. Before the attack starts, there will be a a list $S \subseteq [n]$ of "tamperable" blocks that is *not* necessarily known to the Adv in advance, but will become clear to him as the game goes on. Indeed, this set S itself will be first sampled according to some fixed distribution $\mathbf{S}$, and the crucial condition we require is that $\Pr[i \in \mathbf{S}] = p$ holds for all $i \in [n]$. After $S \leftarrow \mathbf{S}$ is sampled, the sequence of blocks $(x_1, \ldots, x_n)$ will be sampled block-by-block as follows. Assuming (inductively) that $x_1, \ldots, x_{i-1}$ are already sampled so far, if $i \in S$, then Adv gets to fully control x_i and determine its value, but if $i \notin S$, then x_i is simply sampled from its original conditional distribution $(\mathbf{x}_i \mid x_1, \ldots, x_{i-1})$. At the end, the function f is computed over the (adversarially) sampled sequence.

[1] The confidence parameter here is what is usually known as $1 - \delta$ in (ε, δ)-PAC learning, where ε takes the role of our α.

We now explain the intuitive connection between generalized p-tampering attacks and (k, p)-poisoning attacks. The main idea is that we will use a generalized q-tampering attack for $q = p \cdot k/m$ over the random process that lists the sequence of training data provided by the parties during the protocol. Let $\mathbf{S}$ be the distribution over $[n]$ that picks its members through the following algorithm. First choose a set of random parties $\{Q_1, \ldots, Q_k\} \subseteq \{P_1, \ldots, P_m\}$, and then for each message x_j that belongs to Q_i, include the corresponding index j in the final sampled $S \leftarrow \mathbf{S}$ with independent probability p. It is easy to see that $\mathbf{S}$ eventually picks every message with (marginal) probability $q = p \cdot k/m$, but it is also the case that these inclusions are not independent events. Finally, to use the power of generalized p-tampering attacks over the described $\mathbf{S}$ and the random process of messages coming from the parties to get the results of Theorem 1.1, roughly speaking, we let a function f model the loss function applied over the produced hypothesis. Therefore, to prove Theorem 1.1 it is sufficient to prove Theorem 1.2 below which focuses on the power of generalized p-tampering biasing attacks.

Theorem 1.2 (Power of generalized p-tampering attacks–**informal**). *Suppose $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is a joint distribution such that, given any prefix, the remaining blocks could be efficiently sampled in polynomial time. Also let $f\colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto [0, 1]$, and let μ, ν in order be the expected value and variance of $f(\overline{\mathbf{x}})$. Then, for any set distribution $\mathbf{S}$ for which $\Pr[i \in \mathbf{S}] = p$ for all i, there is a polynomial-time generalized p-tampering attack (over tampered blocks in $\mathbf{S}$) that increases the average of f over its input from μ to $\mu' \approx \mu + p \cdot \nu/2$.*

(The formal statement of Theorem 1.2 above follows from Theorem 3.6 and Lemma 3.7.)

Remark 1.3. It is easy to see that in the definition of generalized p-tampering attacks, it does not matter whether we define the attack bit-by-bit or block-by-block. The reason is that, even if we break down each block into smaller bits, then still each bit shall eventually fall into the set of tamperable bits, and the model allows correlation between the inclusion and exclusion of each block/bit into the final tamperable set. This is in contrast to the p-tampering model for which this equivalence is not true. In fact, optimal bounds achievable by bitwise p-tampering as proved in [ACM+17] are *impossible* to achieve in the blockwise p-tampering setting [MM17]. Despite this simplification, we still prefer to use a blockwise presentation of the random process, as this way of modeling the problem allows better tracking measures for the attacker's sample complexity.

Before describing ideas behind the proof of Theorem 1.2 above and its formalization through Theorem 3.6 and Lemma 3.7, we discuss some related work.

Related previous work. Any biasing attack can also be interpreted as some form of impossibility result for a set of imperfect sources that started with [SV86] and expanded to more complicated models (e.g., see [SV86, RVW04, DOPS04, CG88, Dod01, DR06, BEG17, DY15, BBEG18]), and in that regard, our work is no exception. In particular, the work of Beigi, Etesami, and Gohari [BEG17] defined the notion of *generalized* SV sources that generalized SV sources but, due to the correlated nature of tamperable blocks in our model, the two models are incomparable.

In addition, as explained above in more detail, our work is related to the area of attacks on coin-tossing protocols (e.g., see [LLS89, CI93, GKP15, MPS10, CI93, BHT14, HO14, BHLT17, BHT18]), however in the following we discuss some of these works in more depth. Perhaps, the most relevant is the model posed in the open questions section of Lichtenstein, Linial, and Saks [LLS89]. They ask about the power of biasing attackers in coin flipping protocols in which the adversary has a bounded budget b, and after using this budget on any party/message, the corruption will indeed happen with probability p. The main difference to our model is that the adversary does not get to pick the exact tampering locations in our model (which makes

our results stronger). Moreover, in their model each party sends exactly one message, and the attacker can corrupt the parties in an *adaptive* way. Despite the similarities, the difference in the two models makes our results not tight for their setting. Finally, the work of Dodis [Dod01] also defines a tampering model that bears similarities to the generalized p-tampering model. In the model of [Dod01] again an adversary has a bounded budget b to use in its corruption, but even when he does *not* use his budget, he can still try to tamper the parties/messages and *still* succeed with probability p. The latter makes the model of [Dod01] quite different from ours.

1.1 Ideas Behind Our Generalized p-Tampering Attack

Since generalized p-tampering already bears similarities to the model of p-tampering attacks, our starting point is the blockwise p-tampering attack of [MM17]. It was shown in [MM17] that, by using a so-called "one-rejection sampling" (1RS) attack, the adversary can achieve the desired bias of $\Omega(p \cdot \nu)$. So, here we recall the 1RS attack of [MM17].

- In the 1RS attack, for any prefix of already sampled blocks $(x_1, \ldots, x_{i-1})$, suppose the adversary is given the chance of controlling the next i'th block. In that case, the 1RS adversary first samples a full random continuation $x'_i, \ldots, x'_n$ from the marginal distribution of the remaining blocks conditioned on $(x_1, \ldots, x_{i-1})$. Let $s = f(x_1, \ldots, x_{i-1}, x'_i, \ldots, x'_n)$. Then, the 1RS attack keeps the sample x'_i *with probability* s, and changes that into a new fresh sample x''_i with probability $1 - s$.

The great thing about the 1RS attack is that it is already polynomial time assuming that we give access to a sampling oracle that provides the adversary with a random continuation for any prefix. However, the problem with the above attack is that its inductive analysis of [MM17] crucially depends on the tampering probabilities of each block to be *independent* of each other.

The next idea is to modify the 1RS attack of [MM17] based on attacks in the context of coin-tossing protocols and, in particular, the two works of Ben-Or and Linial [BOL89] and Haitner and Omri [HO14]. Indeed, in [BOL89] it was shown that, if the adversary corrupts k parties in an interactive coin-flipping protocol, then it is indeed able to increase the probability of obtaining 1 as follows. Let $\mu = \Pr[\text{output bit} = 1]$, and for a subset $S \subseteq [m]$ of size $|S| = k$ of the parties, let μ_S be the probability of output being 1, if the parties in S use their "optimal" strategy (which is not polynomial-time computable). Then, the result of [BOL89] could be interpreted as follows:

$$\mathsf{GMean}\{\mu_S \mid S \subseteq [m], |S| = k\} \geq \mu^{1-k/m} \tag{1}$$

where GMean denotes the geometric mean (of the elements in the multi-set). Then, by an averaging argument one can show that *there is* at least one set $S \subseteq [m]$ of size $|S| = k$ such that corrupting players in S and using their optimal strategy can achieve expected value at least $\mu^{1-k/m}$, and the bias $\mu^{1-k/m} - \mu$ is indeed $\Omega(k \cdot \mu \cdot (1-\mu)/m)$, which is large enough. However, the proof of [BOL89] does not give a polynomial time attack and uses a rather complicated induction. So, to make it polynomial time and to even make it handle generalized p-tampering attacks, we use one more idea from the follow up work of [HO14]. In [HO14], it was shown that for the case of *two* parties, the biasing bound proved in [BOL89] could be achieved by the following simple (repeated) rejection sampling (RRS) strategy.

- In the RRS attack, for any prefix of already sampled blocks $(x_1, \ldots, x_{i-1})$, suppose the adversary is given the chance of controlling the next i'th block. The good thing about the RRS attack is that it achieves the bounds of the [BOL89] while it could *also* can be made polynomial time in any model where random continuation can be done efficiently. The RRS tampering then works as follows:

1. Let $x'_i, \ldots, x'_n$ be a random continuation of the random process.
2. If $s = f(x_1, \ldots, x_{i-1}, x'_i, \ldots, x'_n)$, then with probability s output y_i , and with probability $1 - s$ go to Step 1 and repeat the sampling process again.

Indeed, we shall point out that for the Boolean case of [BOL89, HO14], the probability s is either zero or one, and the above RRS attack is the adaptation of rejection sampling to the *real-output* case, as done in the context of p-tampering.

Putting things together, our main contributions are taking the following steps to prove Theorem 1.2.

1. We show that the RRS attack of [HO14] does indeed extend to the multi-party case. Interestingly, to prove this, we avoid the inductive proofs of both [BOL89, HO14] and give a direct proof based on the arithmetic-mean vs. geometric-mean inequality (Lemma A.1). However, our proof has a down side: we only get a lower bound on the *arithmetic mean* of $\{\mu_S \mid S \subseteq [m], |S| = k\}$ in Inequality 1. But that is good enough for us, as we indeed want to lower bound the bias achieved when we corrupt a *randomly* selected set of k parties, and the arithmetic mean gives exactly that.

2. We show that the above argument extends even to the case of generalized p-tampering when the weights in the arithmetic mean are proportional to the probabilities of choosing each set S. For doing this, we use an idea from [HO14] that analyzes an imaginary attack in which the adversary does the tampering effort over each block, and then we compare this to the arithmetic mean of actual attacks.

3. We show that our proof extends to the *real-output* case, and achieve a bound that generalizes the bound μ^{1-p} of the Boolean case. As pointed out above, the inductive proofs of [BOL89, HO14] seem to be tightly tailored to the Boolean case, but our direct proof based on the AM-GM inequality scales nicely for the real-output RRS attack described above.

   The lower bound, proved only for the arithmetic mean of $\{\mu_S \mid S \subseteq [m], |S| = k\}$, is equal to $\mu' = \mu^{-p} \cdot \mathbb{E}[f(\overline{\mathbf{x}})^{1+p}]$ (see Theorem 3.6). However, it is not clear that the bound μ' is any better than the original μ! Yet, it can be observed that $\mu' \geq \mu$ always holds due to Jensen's inequality. Therefore, a natural tool for *lower bounding* $\mu' - \mu$ is to use lower bounds on the "gap" of Jensen's inequality. Indeed, we use one such result due to [LB17] (see Lemma A.2) and obtain the desired lower bound of $\mu' - \mu \geq \Omega(p \cdot \nu)$ by simple optimization calculations.

2 Multi-party Poisoning Attacks: Definitions and Main Results

Basic probabilistic notation. We use bold font (e.g., $\mathbf{x}, \mathbf{S}, \boldsymbol{\alpha}$) to represent random variables, and usually use same non-bold letters for denoting samples from these distributions. We use $d \leftarrow \mathbf{d}$ to denote the process of sampling d from the random variable $\mathbf{d}$. By $\mathbb{E}[\boldsymbol{\alpha}]$ we mean the expected value of $\boldsymbol{\alpha}$ over the randomness of $\boldsymbol{\alpha}$, and by $\mathbb{V}[\boldsymbol{\alpha}]$ we denote the variance of random variable $\boldsymbol{\alpha}$. We might use a "processed" version of $\boldsymbol{\alpha}$, and use $\mathbb{E}[f(\boldsymbol{\alpha})]$ and $\mathbb{V}[f(\boldsymbol{\alpha})]$ to denote the expected value and variance, respectively, of $f(\boldsymbol{\alpha})$ over the randomness of $\boldsymbol{\alpha}$.

Notation for learning problems. A learning problem $(\mathcal{A}, \mathcal{B}, \mathbf{d}, \mathcal{H}, \mathsf{Loss})$ is specified by the following components. The set $\mathcal{A}$ is the set of possible *instances*, $\mathcal{B}$ is the set of possible *labels*, $\mathbf{d}$ is distribution over $\mathcal{A} \times \mathcal{B}$.[2] The set $\mathcal{H} \subseteq \mathcal{B}^{\mathcal{A}}$ is called the *hypothesis space* or *hypothesis class*. We consider *loss*

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

functions $\mathsf{Loss}\colon \mathcal{B} \times \mathcal{B} \mapsto \mathbb{R}_+$ where $\mathsf{Loss}(b', b)$ measures how different the 'prediction' y' (of some possible hypothesis $h(a) = y'$) is from the true outcome y. We call a loss function *bounded* if it always takes values in $[0, 1]$. A natural loss function for classification tasks is to use $\mathsf{Loss}(b', b) = 0$ if $y = y'$ and $\mathsf{Loss}(b', b) = 1$ otherwise. The *risk* of a hypothesis $h \in \mathcal{C}$ is the expected loss of h with respect to $\mathbf{d}$, namely $\mathsf{Risk}(h) = \mathbb{E}_{(a,b)\leftarrow\mathbf{d}}[\mathsf{Loss}(h(a), b)]$. An *example* s is a pair $s = (a, b)$ where $x \in \mathcal{A}$ and $y \in \mathcal{B}$. An example is usually sampled from a distribution $\mathbf{d}$. A *sample* set (or sequence) $\mathcal{T}$ of size n is a set (or sequence) of n examples. We assume that instances, labels, and hypotheses are encoded as strings over some alphabet such that given a hypothesis h and an instance x, $h(a)$ is computable in polynomial time.

2.1 Basic Definitions for Multi-party Learning and Poisoning

Multi-party learning problems. An m-party learning problem $(\mathcal{D}, \mathcal{H}, \mathsf{Loss})$ is defined similarly to the (single-party) learning problem (without the sets $\mathcal{A}, \mathcal{B}$ denoted explicitly for a reason to be clear shortly), with the following difference. This time, $\mathcal{D} = (\mathbf{d}_1, \ldots, \mathbf{d}_m, \mathbf{d})$ consists of $m + 1$ distributions (possibly all the same) such that party P_i gets samples from $\mathbf{d}_i$, and they jointly want to learn the distribution $\mathbf{d}$. So, each distribution $\mathbf{d}_i$ might have its own instance space $\mathcal{A}_i$ and label space $\mathcal{B}_i$. The loss function Loss is still defined for the specific target test distribution $\mathbf{d}$. Also, even though, $\mathcal{D}$ is an actual *sequence*, for simplicity we sometimes treat it as a set and write statements like $\mathbf{d}_i \in \mathcal{D}$.

Definition 2.1 (Multi-party learning protocols)**.** An m-party learning protocol Π for the m-party learning problem $(\mathcal{D}, \mathcal{H}, \mathsf{Loss})$ consists of an aggregator function G and m (interactive) data providers $\mathcal{P} = \{P_1, \ldots, P_m\}$. For each data provider P_i, there is a distribution $\mathbf{d}_i \in \mathcal{D}$ that models the (honest) distribution of labeled samples generated by P_i, and there is a final (test) distribution $\mathbf{d}$ that $\mathcal{P}, G$ want to learn jointly. The protocol runs in r rounds and at each round, based on the protocol Π, one particular data owner P_i broadcasts a single labeled example $(a, b) \leftarrow \mathbf{d}_i$.[3] In the last round, the aggregator function G maps the transcript of the messages to an output hypothesis $h \in \mathcal{H}$. For a protocol Π designed for a multi-party problem $(\mathcal{D}, \mathcal{H}, \mathsf{Loss})$, we define the following functions.

- The *confidence* function for a given error threshold $\alpha \in [0, 1]$ is defined as

$$\mathsf{Conf}(\alpha, \mathbf{d}) = \Pr_{h\leftarrow\Pi}[\mathsf{Risk}(h, \mathbf{d}) \leq \alpha].$$

- The *average error* (or average loss) for a specific example $d = (a, b) \leftarrow \mathbf{d}$ is defined as

$$\mathsf{Err}(d) = \Pr_{h\leftarrow\Pi}[\mathsf{Loss}(h(a), b)],$$

  based on which the total error of the protocol is defined as $\mathsf{Err}(\mathbf{d}) = \mathbb{E}_{d\leftarrow\mathbf{d}}[\mathsf{Err}(d)]$.

Now, we define poisoning attackers that target multi-party protocols. We formalize a more general notion that covers both p-tampering attackers as well as attackers who (statically) corrupt k parties.

Definition 2.2 (Multi-party (k, p)-poisoning attacks)**.** A (k, p)-poisoning attack against an m-party learning protocol Π is defined by an adversary Adv who can control a subset $\mathcal{C} \subseteq [m]$ of the parties where $|\mathcal{C}| = k$. The attacker Adv shall pick the set $\mathcal{C}$ at the beginning. At each round j of the protocol, if a data provider $P_i \in \mathcal{C}$ is supposed to broadcast the next example from its distribution $\mathbf{d}_i$, the adversary can partially

[3] We can directly model settings where more data is exchanged in one round, however, we stick to the simpler definition as it loses no generality.

control this sample use tampered distribution $\widetilde{\mathbf{d}}$ such that $|\widetilde{\mathbf{d}} - \mathbf{d}_i| \leq p$ in statistical distance. Note that the distribution $\widetilde{\mathbf{d}}$ can depend on the history of examples broadcast so far, but the requirement is that, conditioned on this history, the malicious message of adversary modeled by distribution $\widetilde{\mathbf{d}}$, is at most p-statistically far from $\mathbf{d}_i$. We use Π_{Adv} to denote the protocol in presence of Adv. We also define the following notions.

- We call Adv a *plausible* adversary, if it always holds that $\mathrm{Supp}(\widetilde{\mathbf{d}}) \subseteq \mathrm{Supp}(\mathbf{d}_i)$.
- Adv is *efficient* if it runs in polynomial time in the total length of the messages exchanged during the protocol (from the beginning till end).
- The *confidence* function in presence of Adv is defined as

$$\mathsf{Conf}_{\mathrm{Adv}}(\alpha, \mathbf{d}) = \Pr_{h \leftarrow \Pi_{\mathrm{Adv}}}[\mathsf{Risk}(h, \mathbf{d}) \leq \alpha]$$

  and $\mathsf{Conf}(\alpha, \mathbf{d})$ is the confidence of the learning protocol without any attacks which can be formally defined using an attacker I who does not change any of the distributions.
- The *average error* for a specific example $d = (a, b) \leftarrow \mathbf{d}$ in presence of Adv is defined as

$$\mathsf{Err}_{\mathrm{Adv}}(d) = \Pr_{h \leftarrow \Pi}[\mathsf{Loss}(h(a), b)],$$

  based on which the total error of the protocol is defined as $\mathsf{Err}_{\mathrm{Adv}}(\mathbf{d}) = \mathbb{E}_{d \leftarrow \mathbf{d}}[\mathsf{Err}(d)]$.

Note that standard (non-adversarial) confidence and average error functions could be also defined as adversarial ones using a trivial adversary I who simply outputs its input.

Remark 2.3 (Static vs. adaptive corruption)**.** Definition 2.2 focuses on corrupting k parties statically. A natural extension of this definition in which the set $\mathcal{C}$ is chosen *adaptively* [CFGN96] while the protocol is being executed can also be defined naturally. In this work, however, we focus on static corruption, and leave the possibility of improving our results in the adaptive case for future work.

2.2 Power of Multy-party Poisoning Attacks

We now formally state our result about the power of (k, p)-poisoning attacks.

Theorem 2.4 (Power of efficient multi-party poisoning)**.** *In any m-party protocol Π for parties $\mathcal{P} = \{P_1, \ldots, P_m\}$, for any $p \in [0, 1]$ and $k \in [m]$, the following hold where M is the total length of the messages exchanged during the protocol.*

1. *For any $\alpha, \varepsilon \in [0, 1]$, there is a* plausible*, (k, p)-poisoning attack Adv that runs in time $\mathrm{poly}(M/\varepsilon)$ and decreases the confidence of the protocol as follows*

$$\mathsf{Conf}_{\mathrm{Adv}}(\alpha, \mathbf{d}) \leq \left(1 - p \cdot \frac{k}{m}\right) \cdot \mathsf{Conf}(\alpha, \mathbf{d}) + \varepsilon.$$

2. *If the (normalized) loss function is bounded (i.e., it outputs in $[0, 1]$), then there is a* plausible*, (k, p)-poisoning Adv that runs in time $\mathrm{poly}(M/\varepsilon)$ and increases the average error of the protocol as*

$$\mathsf{Err}_{\mathrm{Adv}}(\mathbf{d}) \geq \mathsf{Err}(\mathbf{d}) + \frac{p \cdot k}{2m} \cdot \nu - \varepsilon$$

   where $\nu = \mathbb{V}_{h \leftarrow \Pi}[\mathsf{Risk}(h, \mathbf{d})]$ (and $\mathbb{V}[\cdot]$ denotes the variance).

3. *If* Loss *is a Boolean function (e.g. as in classification problems), for any final test example* $d \leftarrow \mathbf{d}$*, there is a* plausible*,* (k,p)*-poisoning attack* Adv *that runs in time* $\mathrm{poly}(M/\varepsilon)$ *and increases the average error of the test example* d *as follows,*

$$\mathsf{Err}_{\mathrm{Adv}}(d) \geq \mathsf{Err}(d) + p \cdot \frac{k}{m} \cdot (1 - \mathsf{Err}(d)) - \varepsilon.$$

Before proving Theorem 2.4, we need to develop our main result about the power of generalized p-tampering attacks; in Section 3, we develop such tools, and then in Section 4.1 we prove Theorem 2.4.

Remark 2.5 (Allowing different distributions in different rounds)**.** In Definition 2.2, we restrict the adversary to remain "close" to $\mathbf{d}_i$ for each message sent out by one of the corrupted parties. A natural question is: what happens if we allow the parties distributions to be different in different rounds. For example, in a round j, a party P_i might send *multiple* training examples $D^{(j)} = \left(d_1^{(j)}, d_2^{(j)}, \ldots, d_k^{(j)}\right)$, and we want to limit the *total* statistical distance between the distribution of the larger message $D^{(j)}$ from $\mathbf{d}_i^k$ (i.e., k iid samples from $\mathbf{d}_i$).[4] We emphasize that, our results extend to this more general setting as well. In particular, the proof of Theorem 2.4 directly extends to a more general setting where we can allow the honest distribution $\mathbf{d}_i$ of each party i to also depend on the round j in which these messages are sent. Thus, we can use a round-specific distribution $\mathbf{d}_i^{(j)}$ to model the joint distribution of *multiple* samples $D^{(j)} = \left(d_1^{(j)}, d_2^{(j)}, \ldots, d_k^{(j)}\right)$ that are sent out in the j'th round by the party P_i. This way, we can obtain the stronger form of attacks that remain statistically close to the joint (correct) distribution of the (multi-sample) messages sent in a round. In fact, as we will discuss shortly $D^{(j)}$ might be of completely different type, e.g., just some shared random bits.

Remark 2.6 (Allowing randomized aggregation)**.** The aggregator G is a simple function that maps the transcript of the exchanged messages to a hypothesis h. A natural question is: what happens if we generalize this to the setting where G is allowed to be randomized. We note that in Theorem 2.4, Part 2 can allow G to be randomized, but Parts 1 and 3 need deterministic aggregation. The reason is that for those parts, we need the transcript to determine the confidence and average error functions. One general way to make up for randomized aggregation is to allow the parties to inject randomness into the transcript as they run the protocol by sending messages that are not necessarily learning samples from their distribution $\mathbf{d}_i$. As described in Remark 2.5, our attacks extend to this more general setting as well. Otherwise, we will need the adversary to be able to also depend on the randomness of G, but that is also a reasonable assumption if the aggregation is used using public beacon that could be obtained by the adversary as well.

3 Generalized p-Tampering Biasing Attacks: Definitions and Main Results

In this section, we formally state our main result about the power of generalized p-tampering attacks. we start by formalizing some notation and basic definitions.

3.1 Preliminary Notation and Basic Definitions for Tampering with Random Processes

Notation. By $\mathbf{x} \equiv \mathbf{y}$ we denote that the random variables $\mathbf{x}$ and $\mathbf{y}$ have the same distributions. Unless stated otherwise, by using a bar over a variable, we emphasize that it is a vector. By $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ we refer to a joint distribution over vectors with n components. For a joint distribution $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, we use $\mathbf{x}_{\leq i}$ to denote the joint distribution of the first i variables $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_i)$. Also, for a vector

[4] Note that, even if each block in $\left(d_1^{(j)}, d_2^{(j)}, \ldots, d_k^{(j)}\right)$ remains p-close to $\mathbf{d}_i$, their joint distribution could be quite far from $\mathbf{d}_i^k$.

$\overline{x} = (x_1 \dots x_n)$ we use $x_{\leq i}$ to denote the prefix $(x_1, \dots, x_i)$. For a randomized algorithm $L(\cdot)$, by $y \leftarrow L(x)$ we denote the randomized execution of L on input x outputting y. For a distribution $(\mathbf{x}, \mathbf{y})$, by $(\mathbf{x} \mid \mathbf{y})$ we denote the conditional distribution $(\mathbf{x} \mid \mathbf{y} = y)$. By $\mathrm{Supp}(\mathbf{d}) = \{d \mid \Pr[\mathbf{d} = d] > 0\}$ we denote the support set of $\mathbf{d}$. By $T^{\mathbf{d}}(\cdot)$ we denote an algorithm $T(\cdot)$ with oracle access to a sampler for $\mathbf{d}$ that upon every query returns fresh samples from $\mathbf{d}$. By $\mathbf{d}^n$ we denote the distribution that returns n iid samples from $\mathbf{d}$.

Definition 3.1 (Valid prefixes)**.** Let $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \dots, \mathbf{x}_n)$ be an arbitrary joint distribution. We call $x_{\leq i} = (x_1, \dots, x_i)$ a *valid prefix* for $\overline{\mathbf{x}}$ if there exist $x_{i+1}, \dots, x_n$ such that $(x_1, \dots, x_n) \in \mathrm{Supp}(\overline{\mathbf{x}})$. $\mathrm{ValPref}(\overline{\mathbf{x}})$ denotes the set of all valid prefixes of $\overline{\mathbf{x}}$.

Definition 3.2 (Tampering with random processes)**.** Let $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \dots, \mathbf{x}_n)$ be an arbitrary joint distribution. We call a (potentially randomized and possibly computationally unbounded) algorithm T an (online) *tampering algorithm for* $\overline{\mathbf{x}}$ if given any valid prefix $x_{\leq i-1} \in \mathrm{ValPref}(\overline{\mathbf{x}})$, it holds that

$$\Pr_{x_i \leftarrow \mathsf{T}(x_{\leq i-1})}[x_{\leq i} \in \mathrm{ValPref}(\overline{\mathbf{x}})] = 1 \ .$$

Namely, $\mathsf{T}(x_{\leq i-1})$ outputs x_i such that $x_{\leq i}$ is again a valid prefix. We call T an *efficient* tampering algorithm for $\overline{\mathbf{x}}$ if it runs in time $\mathrm{poly}(N)$ where N is maximum bit length to represent any $\overline{x} \in \mathrm{Supp}(\overline{\mathbf{x}})$.

Definition 3.3 (Online samplers)**.** We call OnSam an *online sampler* for $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \dots, \mathbf{x}_n)$ if for all $x_{\leq i-1} \in \mathrm{ValPref}(\overline{\mathbf{x}})$, $\mathsf{OnSam}(n, x_{\leq i-1}) \equiv \mathbf{x}_i$. Moreover, we call $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \dots, \mathbf{x}_n)$ *online samplable* if it has an online sampler that runs in time $\mathrm{poly}(N)$ where N is maximum bit length of any $\overline{x} \in \mathrm{Supp}(\overline{\mathbf{x}})$.

Definition 3.4 (Notation for tampering distributions)**.** Let $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \dots, \mathbf{x}_n)$ be an arbitrary joint distribution and T a tampering algorithm for $\overline{\mathbf{x}}$. For any subset $S \subseteq [n]$, we define $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{T}, S \rangle$ to be the joint distribution that is the result of online tampering of T over set S, where $\overline{\mathbf{y}} \equiv (\mathbf{y}_1, \dots, \mathbf{y}_n)$ is sampled inductively as follows. For every $i \in [n]$, suppose $y_{\leq i-1}$ is the previously sampled block. If $i \in S$, then the i^{th} block $\mathbf{y}_i$ is generated by the tampering algorithm $\mathsf{T}(y_{\leq i-1})$, and otherwise, $\mathbf{y}_i$ is sampled from $(\mathbf{x}_i \mid \mathbf{x}_{i-1} = y_{\leq i-1})$. For any *distribution* $\mathbf{S}$ over subsets of $[n]$, by $\langle \overline{\mathbf{x}} \parallel \mathsf{T}, \mathbf{S} \rangle$ we denote the random variable that can be sampled by first sampling $S \leftarrow \mathbf{S}$ and then sampling $\overline{y} \leftarrow \langle \overline{\mathbf{x}} \parallel \mathsf{T}, S \rangle$.

3.2 Power of Generalized p-Tampering Attacks

Having the definitions above, we finally describe our main result about the power of generalized p-tampering attacks. We first formalize the way tampering blocks are chosen in such attacks.

Definition 3.5 (p-covering)**.** Let $\mathbf{S}$ be a distribution over the subsets of $[n]$. We call $\mathbf{S}$ a *p-covering* distribution on $[n]$ (or simply p-covering, when n is clear from the context), if for all $i \in [n]$, $\Pr_{S \leftarrow \mathbf{S}}[i \in S] = p$.

Theorem 3.6 (Biasing of bounded functions through generalizing p-tampering)**.** *Let $\mathbf{S}$ be a p-covering distribution on $[n]$, $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \dots, \mathbf{x}_n)$ be a joint distribution, $f\colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto [0, 1]$, and $\mu = \mathbb{E}[f(\overline{\mathbf{x}})]$. Then,*

1. **Computationally unbounded attack.** *There is a (computationally unbounded) tampering algorithm T such that if we let $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{T}, \mathbf{S} \rangle$ be the tampering distribution of T over $S \leftarrow \mathbf{S}$, then*

$$\mathbb{E}[f(\overline{\mathbf{y}})] \geq \mu^{-p} \cdot \mathbb{E}\left[f(\overline{\mathbf{x}})^{1+p}\right] \ .$$

2. **Polynomial-time attack.** *For any $\varepsilon \in [0, 1]$, there exists a tampering algorithm T_ε that, given oracle access to f and any online sampler OnSam for $\overline{\mathbf{x}}$, it runs in time $\mathrm{poly}(N/\varepsilon)$, where N is the bit length of any $\overline{x} \leftarrow \overline{\mathbf{x}}$, and for $\overline{\mathbf{y}}_\varepsilon \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{T}_\varepsilon^{f, \mathsf{OnSam}}, \mathbf{S} \rangle$, it holds that*

$$\mathbb{E}\left[f(\overline{\mathbf{y}}_\varepsilon)\right] \geq \mu^{-p} \cdot \mathbb{E}\left[f(\overline{\mathbf{x}})^{1+p}\right] - \varepsilon \ .$$

Special case of Boolean functions. When the function f is Boolean, we get $\mu^{-p} \cdot \mathbb{E}[f(\overline{\mathbf{x}})^{1+p}] = \mu^{1-p} \geq \mu(1+\Omega_\mu(p))$, which matches the bound proved in [BOL89] for the special case of $p = k/n$ for integer $k \in [n]$ and for $\mathbf{S}$ that is uniformly random subset of $[n]$ of size k. (The same bound for the case of 2 parties was proved in [HO14] with extra properties). Even for this case, compared to [BOL89, HO14] our result is more general, as we can allow $\mathbf{S}$ with arbitrary $p \in [0,1]$ *and* achieve a polynomial time attack given oracle access to an online sampler for $\overline{\mathbf{x}}$. The work of [HO14] also deals with polynomial time attackers for the special case of 2 parties, but their efficient attackers use a different oracle (i.e., OWF inverter), and it is not clear whether or not their attack extend to the case of more then 2 parties. Finally, both [BOL89, HO14] prove their bound for the *geometric* mean of the averages for different $S \leftarrow \mathbf{S}$, while we do so for their arithmetic mean, but we emphasize that this is enough for all of our applications.

The bounds of Theorem 3.6 for both cases rely on the quantity $\mu' = \mu^{-p} \cdot \mathbb{E}[f(\overline{\mathbf{x}})^{1+p}]$. A natural question is: how large is μ' compared to μ? As discussed above, for the case of Boolean f, we already know that $\mu' \geq \mu$, but that argument does not apply to the real-output f. A simple application of Jensen's inequality shows that $\mu \leq \mu'$ in general, but that still does not mean that $\mu' \gg \mu$ (i.e., that there is a large enough gap).

General case of real-output functions: relating the bias to the variance. If $\mathbb{V}[f(\overline{\mathbf{x}})] = 0$, then no tampering attack can achieve any bias, so any gap achieved between μ' and μ shall somehow depend on the variance of $f(\overline{\mathbf{x}})$. In the following, we show that this gap does exist and that $\mu' - \mu \geq \Omega(p \cdot \mathbb{V}[f(\overline{\mathbf{x}})])$. similar results (relating the bias the the variance of the original distribution) were previously proved [MDM18, MM17, ACM+14] for the special case of p-tampering attacks (i.e., $\mathbf{S}$ chooses every $i \in [n]$ independently with probability p). Here we obtain a more general statement that holds for any p-covering set structure $\mathbf{S}$.

Using Lemma 3.7 below for $\boldsymbol{\alpha} \equiv f(\overline{\mathbf{x}})$, we immediately get $\Omega(p \cdot \mathbb{V}[f(\overline{\mathbf{x}})])$ lower bounds for the bias achieved by (both versions of) the attackers of Theorem 3.6 for the general case of real-valued functions and arbitrary p-covering set distribution $\mathbf{S}$.

Lemma 3.7. *Let $\boldsymbol{\alpha}$ be any real-valued random variable over $\mathrm{Supp}(\boldsymbol{\alpha}) \subseteq [0,1]$, and $p \in [0,1]$. Let $\mu = \mathbb{E}[\boldsymbol{\alpha}]$ be the expected value of $\boldsymbol{\alpha}$, $\nu = \mathbb{V}[\boldsymbol{\alpha}]$ be the variance of $\boldsymbol{\alpha}$, and $\gamma = \mu^{-p} \cdot \mathbb{E}[\boldsymbol{\alpha}^{1+p}] - \mu$. Then, it holds that*

$$\gamma \geq \frac{p \cdot (p+1)}{2 \cdot \mu^p} \cdot \nu \geq \frac{p}{2} \cdot \nu \,. \tag{2}$$

4 Proofs of the Main Results

In the following subsections, we will first prove Theorem 2.4 using Theorem 3.6 and Lemma 3.7, and then we will prove Theorem 3.6 and Lemma 3.7.

4.1 Obtaining (k,p)-Poisoning Attacks: Proof of Theorem 2.4

In this subsection, we formally prove Theorem 2.4 using Theorems 3.6 and Lemma 3.7.

For a subset $C \subseteq [m]$ let $P_C = \{P_i; i \in C\}$ and R_C be the subset of rounds where one of the parties in P_C sends an example. Also for a subset $S \subseteq [n]$, we define $\mathbf{Bion}(S,p)$ to be a distribution over all the subsets of S, where each subset $S' \subseteq S$ hast the probability $p^{|S'|} \cdot (1-p)^{|S|-|S'|}$. Now, consider the covering $\mathbf{S}$ of the set $[n]$ which is distributed equivalent to the following process. First sample a uniform subset C of $[m]$ of size k. Then sample and output a set S sampled from $\mathbf{Bion}(R_C, p)$. $\mathbf{S}$ is clearly a $(p \cdot \frac{k}{m})$-covering. We will use this covering to prove all the three statements of the theorem. Before proving the statement we define several notions. For $j \in [n]$ let $w(j)$ be the index of the provider at round j and let $\mathbf{d}_{w(j)}$ be

the designated distribution of the jth round and let $\overline{\mathbf{x}} = \mathbf{d}_{w(1)} \times \cdots \times \mathbf{d}_{w(n)}$. Now we prove the first part of the theorem. We define a function $f_1 : \mathrm{Supp}(\overline{\mathbf{x}}) \to \{0,1\}$, which is a Boolean function and is 0 if the output of the protocol has risk less than or equal to ε and 1 otherwise (note that f is equivalent to $1 - \mathsf{Conf}$). Note that this function f_1 can be approximated in polynomial time, however, here for simplicity, we are assuming that it can be exactly computed in polynomial time. Now we use Theorem 3.6. We know that $\mathbf{S}$ is a $(p \cdot \frac{k}{m})$-covering for $[n]$. Therefore by Part 2 of Theorem 3.6, there exist an $\mathrm{poly}(M/\varepsilon)$ time tampering algorithm T_ε that changes $\overline{\mathbf{x}}$ to $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{T}_\varepsilon^{f_1,\mathsf{OnSam}}, \mathbf{S} \rangle$ where

$$\mathbb{E}[f(\overline{\mathbf{y}})] \geq \mu_1^{-p\cdot k/m} \cdot \mathbb{E}[f(\overline{\mathbf{x}})^{1+p\cdot k/m}] - \varepsilon = \mu_1^{1-p\cdot k/m} - \varepsilon \geq \mu_1 + p \cdot \frac{k}{m} \cdot (1-\mu) - \varepsilon.$$

By an averaging argument, we can conclude that there exist a set $C \in [m]$ of size k for which the distribution $\mathbf{Bion}(R_C, p)$ produces a bias at least $p \cdot \frac{k}{m} \cdot (1-\mu)$. Note that the measure of empty set in $\mathbf{Bion}(R_C, p)$ is exactly equal to $1-p$ which means with probability $1-p$ the adversary will not tamper with any of the blocks, therefore, the statistical distance $|\overline{\mathbf{x}} - \langle \overline{\mathbf{x}} \parallel \mathsf{T}_\varepsilon^{f_1,\mathsf{OnSam}}, \mathbf{Bion}(R_C, p)\rangle|$ is at most p. This concludes the proof of the first part.

Now we prove the second part. The second part is very similar to first part except that the function that we define here is a real valued function. Consider the function $f_2 : \mathrm{Supp}(\overline{\mathbf{x}}) \to [0,1]$ which is defined to be the risk of the output hypotheses. Now by Theorem 3.6 and Lemma 3.7, we know that there is tampering algorithm T_ε that changes $\overline{\mathbf{x}}$ to $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{T}_\varepsilon^{f_2,\mathsf{OnSam}}, \mathbf{S} \rangle$ such that

$$\mathbb{E}[f_2(\overline{\mathbf{y}})] \geq \mu_2 + \frac{p \cdot k}{2m} \cdot \nu.$$

By a similar averaging argument we can conclude the proof.

Now we prove Part 3. Again we define a Boolean function $f_3 : \mathrm{Supp}(\overline{\mathbf{x}}) \to \{0,1\}$ which outputs the loss of the final hypothesis on the example d. Note that f_3 is Boolean since the loss function is Boolean. f_3 is also computable by the adversary because he knows the target example d. Again by a similar use of Theorem 3.6 and averaging argument we can conclude the proof.

4.2 Computationally Unbounded Attacks: Proving Part 1 of Theorem 3.6

Construction 4.1 (Rejection-sampling tampering)**.** Let $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and $f\colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto [0,1]$. The *rejection sampling* tampering algorithm RejSam^f works as follows. Given the valid prefix $y_{\leq i-1} \in \mathrm{ValPref}(\overline{\mathbf{x}})$, the tampering algorithm would do the following:

1. Sample $y_{\geq i} \leftarrow (\mathbf{x}_{\geq i} \mid y_{\leq i-1})$ by using the online sampler for f.
2. If $s = f(y_1, \ldots, y_n)$, then with probability s output y_i, otherwise go to Step 1 and repeat the process.

We will first prove a property of the rejection sampling algorithm when applied on every block.

Definition 4.2 (Notation for partial expectations of functions)**.** Suppose $f\colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto \mathbb{R}$ is defined over a joint distribution $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, $i \in [n]$, and $x_{\leq i} \in \mathrm{ValPref}(\overline{\mathbf{x}})$. Then, using a small hat, we define the notation $\hat{f}(x_{\leq i}) = \mathbb{E}_{\overline{x} \leftarrow (\overline{\mathbf{x}} | x_{\leq i})}[f(\overline{x})]$. (In particular, for $\overline{x} = x_{[n]}$, we have $\hat{f}(\overline{x}) = f(\overline{x})$.)

Claim 4.3. *If* $\langle \overline{\mathbf{x}} \parallel \mathsf{RejSam}^f, [n] \rangle \equiv \overline{\mathbf{y}}^{[n]} \equiv (\mathbf{y}_1, \ldots, \mathbf{y}_n)$*. Then, for every valid prefix* $y_{\leq i} \in \mathrm{ValPref}[\overline{\mathbf{x}}]$*,*

$$\frac{\Pr[\mathbf{y}_{\leq i} = y_{\leq i}]}{\Pr[\mathbf{x}_{\leq i} = y_{\leq i}]} = \frac{\hat{f}(y_{\leq i})}{\mu}.$$

Proof. Based on the description of RejSam^f, for any $y_{\le i} \in \mathrm{ValPref}(\overline{\mathbf{x}})$ the following equation holds for the probability of sampling y_i conditioned on prefix $y_{\le i-1}$.

$$\Pr[\mathbf{y}_i = y_i \mid y_{\le i-1}] = \Pr[\mathbf{x}_i = y_i \mid y_{\le i-1}] \cdot \hat{f}(y_{\le i}) + (1 - \hat{f}(y_{\le i-1})) \cdot \Pr[\mathbf{y}_i = y_i \mid y_{\le i-1}].$$

The first term in this equation corresponds to the probability of selecting and accepting in the first round of sampling and the second term corresponds to the probability of selecting and accepting in any round except the first round. Therefore we have

$$\Pr[\mathbf{y}_i = y_i \mid y_{\le i-1}] = \frac{\hat{f}(y_{\le i})}{\hat{f}(y_{\le i-1})} \cdot \Pr[\mathbf{x}_i = y_i \mid y_{\le i-1}] \,,$$

which implies that

$$\Pr[\mathbf{y}_{\le i} = y_{\le i}] = \prod_{j \in [i]} \left(\frac{\hat{f}(y_{\le j})}{\hat{f}(y_{\le j-1})} \right) \cdot \Pr[\mathbf{x}_{\le i} = y_{\le i}] = \frac{\hat{f}(y_{\le i})}{\mu} \cdot \Pr[\mathbf{x}_{\le i} = y_{\le i}] \,.$$

□

Now, we prove two properties for *any* tampering algorithm (not just rejection sampling) over a p-covering set distribution.

Lemma 4.4. *Let $\mathbf{S}$ be p-covering for $[n]$ and $\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})$. For any $S \in \mathrm{Supp}(\mathbf{S})$ and an arbitrary tampering algorithm T for $\overline{\mathbf{x}}$, let $\overline{\mathbf{y}}^S \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{T}, S \rangle$. Then,*

$$\prod_{S \in 2^{[n]}} \left(\frac{\Pr[\overline{\mathbf{y}}^S = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \right)^{\Pr[\mathbf{S}=S]} = \left(\frac{\Pr[\overline{\mathbf{y}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \right)^p \,.$$

Proof. For every $y_{\le i} \in \mathrm{ValPref}(\overline{\mathbf{y}}^{[n]}) \subseteq \mathrm{ValPref}(\overline{\mathbf{x}})$ define $\rho[y_{\le i}]$ as

$$\rho[y_{\le i}] = \frac{\Pr[\mathbf{y}_i^{[n]} = x_i \mid \mathbf{y}_{\le i-1}^{[n]} = y_{\le i-1}]}{\Pr[\mathbf{x}_i = x_i \mid \mathbf{x}_{\le i-1} = y_{\le i-1}]} \,.$$

Then, for all $\overline{y} \in \mathrm{ValPref}(\overline{\mathbf{y}}^S) \subseteq \mathrm{ValPref}(\overline{\mathbf{x}})$ we have

$$\Pr[\overline{\mathbf{y}}^S = y] = \Pr[\overline{\mathbf{x}} = y] \cdot \prod_{i \in S} \rho[y_{\le i}] \,.$$

Therefore,

$$\prod_{S \in 2^{[n]}} \left(\frac{\Pr[\overline{\mathbf{y}}^S = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \right)^{\Pr[\mathbf{S}=S]} = \prod_{S \in 2^{[n]}} \left(\prod_{i \in S} \rho[y_{\le i}] \right)^{\Pr[\mathbf{S}=S]} = \prod_{i \in [n]} \left(\rho[y_{\le i}] \right)^{\Pr[i \in \mathbf{S}]} = \left(\prod_{i \in [n]} \rho[y_{\le i}] \right)^p \,.$$

□

Claim 4.5. *Suppose $\mathbf{S}$ is p-covering on $[n]$, $\overline{\mathbf{y}}^S \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{T}, S \rangle$ for any $S \leftarrow \mathbf{S}$, and $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{T}, \mathbf{S} \rangle$ for an arbitrary tampering algorithm T for $\overline{\mathbf{x}}$. Then, it holds that*

$$\mathbb{E}[f(\overline{\mathbf{y}})] \ge \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \cdot \left(\frac{\Pr[\overline{\mathbf{y}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \right)^p \,.$$

Proof. Note that $\mathrm{Supp}(\overline{\mathbf{y}}^S) \subseteq \mathrm{Supp}(\overline{\mathbf{x}})$ for any $S \subseteq [n]$. Therefore,

$$\begin{aligned}
\mathbb{E}[f(\overline{\mathbf{y}})] = \mathop{\mathbb{E}}_{S \leftarrow \mathbf{S}} \mathop{\mathbb{E}}_{\overline{y} \leftarrow \overline{\mathbf{y}}^S}[f(y)] &= \sum_{S \in 2^{[n]}} \Pr[\mathbf{S} = S] \cdot \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \Pr[\overline{\mathbf{y}}^S = \overline{y}] \cdot f(\overline{y}) \\
&= \sum_{S \in 2^{[n]}} \Pr[\mathbf{S} = S] \cdot \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \frac{\Pr[\overline{\mathbf{y}}^S = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\
&= \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \cdot \sum_{S \in 2^{[n]}} \Pr[\mathbf{S} = S] \cdot \left(\frac{\Pr[\overline{\mathbf{y}}^S = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \right) \\
\text{(by AM-GM inequality of Lemma A.1)} \quad &\geq \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \cdot \prod_{S \in 2^{[n]}} \left(\frac{\Pr[\overline{\mathbf{y}}^S = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \right)^{\Pr[\mathbf{S}=S]} \\
\text{(by } p\text{-covering of } \mathbf{S} \text{ and Lemma 4.4)} \quad &= \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \cdot \left(\frac{\Pr[\overline{\mathbf{y}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \right)^{p}.
\end{aligned}$$

□

We now prove the main result using the one-rejection sampling tampering algorithm and also relying on the p-covering property of $\mathbf{S}$. In particular, if $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{RejSam}^f, \mathbf{S} \rangle$, then by Claims 4.5 and 4.3 we have

$$\begin{aligned}
\mathbb{E}[f(\overline{\mathbf{y}})] &\geq \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{\Pr[\overline{\mathbf{y}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]} \right)^{p} \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\
\text{(by Claim 4.3)} \quad &= \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu} \right)^{p} \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\
&= \mu^{-p} \cdot \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y})^{1+p} = \mu^{-p} \cdot \mathbb{E}[f(\overline{\mathbf{x}})^{1+p}] .
\end{aligned}$$

4.3 Polynomial-time Attacks: Proving Part 2 of Theorem 3.6

In this section, we prove the second item of Theorem 3.6. Namely, we show an efficient tampering algorithm whose average is ε-close to the average of RejSam. We define this attack as follows:

Construction 4.6 (k-rejection-sampling tampering)**.** Let $\overline{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a joint distribution and $f\colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto [0,1]$. The *$k$-rejection sampling* tampering algorithm RejSam_k^f works as follows. Given the valid prefix $y_{\leq i-1} \in \mathrm{ValPref}(\overline{\mathbf{x}})$, the tampering algorithm would do the following for k times:

1. Sample $y_{\geq i} \leftarrow (\mathbf{x}_{\geq i} \mid y_{\leq i-1})$ by using the online sampler for f.
2. Let $s = f(y_1, \ldots, y_n)$; with probability s output y_i, otherwise go to Step 1.

If no y_i was output during any of the above k iterations then output a fresh sample $y_i \leftarrow (\mathbf{x}_i \mid y_{\leq i-1})$.

The output distribution of RejSam_k on any input, converges to the rejections sampling tampering algorithm RejSam for sufficiently large $k \to \infty$.

Notation. Below, use the notation $\overline{\mathbf{z}} = \langle \overline{\mathbf{x}} \parallel \mathsf{RejSam}_k^f, S \rangle$ and $\mu_k = \mathbb{E}[f(\overline{\mathbf{z}})]$.

We will prove the following claim which will directly completes the proof of second part of Theorem 3.6.

Claim 4.7. *Let $\overline{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a joint distribution and $f\colon \mathrm{Supp}(\overline{\mathbf{x}}) \mapsto [0,1]$. For any $\varepsilon \in [0,1]$, let $k \geq \frac{16 \ln(2n/\varepsilon)}{\varepsilon^2 \mu^2}$. Then RejSam_k runs in time $O(k) = \mathrm{poly}(N/(\varepsilon \cdot \mu))$, where $N \geq n$ is the total bit-length of representing $\overline{\mathbf{x}}$, and for $\overline{\mathbf{z}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{RejSam}_k^{f,\mathsf{OnSam}}, \mathbf{S} \rangle$ it holds that*

$$\mathbb{E}[f(\overline{\mathbf{z}})] \geq \mu^{-p} \cdot \mathbb{E}[f(\overline{\mathbf{x}})^{1+p}] - \varepsilon \, .$$

Proof. It is easy to see why RejSam_k runs in time $O(k)$ and thus we will focus on proving the expected value of the output of the k-rejection sampling tampering algorithm. To that end, we start by providing some definitions relevant to our analysis.

Definition 4.8. For $\delta \geq 0$, let

$$\mathrm{High}(\delta) = \{\overline{x} \mid \overline{x} \in \mathrm{Supp}(\overline{\mathbf{x}}) \wedge \forall i \in [n], \hat{f}(x_{\leq i-1}) \geq \delta\}, \ \mathrm{Low}(\delta) = \mathrm{Supp}(\overline{\mathbf{x}}) \setminus \mathrm{High}(\delta) \, ,$$

$$\mathrm{Big}(\delta) = \{\overline{x} \mid \overline{x} \in \mathrm{Supp}(\overline{\mathbf{x}}) \wedge f(\overline{x}) \geq \delta\}, \text{ and } \ \mathrm{Small}(\delta) = \mathrm{Supp}(\overline{\mathbf{x}}) \setminus \mathrm{Big}(\delta) \, .$$

Claim 4.9. *For $\delta_1 \cdot \delta_2 = \delta$, it holds that*

$$\Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \mid \overline{x} \in \mathrm{Low}(\delta)] \leq \delta_2 \, .$$

As a result, it holds that $\Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \wedge \overline{x} \in \mathrm{Low}(\delta)] \leq \delta_2$, and so

$$\sum_{\overline{x} \in \mathrm{Big}(\delta_1) \cap \mathrm{Low}(\delta)} \Pr[\overline{x} = \overline{\mathbf{x}}] \leq \delta_2 \, .$$

Proof. Let $t\colon \mathrm{Low}(\delta) \to \mathrm{ValPref}(\overline{\mathbf{x}})$ be such that $t(\overline{x})$ is the smallest prefix $x_{\leq i}$ such that $\hat{f}(x_{\leq i}) \leq \delta$. Now consider the set $T = \{t(\overline{x}) \mid \overline{x} \in \mathrm{Low}(\delta)\}$. For any $w \in T$ we have

$$\delta \geq \hat{f}(w) \geq \Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \mid t(\overline{x}) = w] \cdot \delta_1 \, ,$$

which implies

$$\Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \mid t(\overline{x}) = w] \leq \delta_2 \, .$$

Thus, we have

$$\begin{aligned}
&\Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \mid \overline{x} \in \mathrm{Low}(\delta)] \\
&= \sum_{w \in T} \Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \wedge t(\overline{x}) = w \mid \overline{x} \in \mathrm{Low}(\delta)] \\
&= \sum_{w \in T} \Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[\overline{x} \in \mathrm{Big}(\delta_1) \mid \overline{x} \in \mathrm{Low}(\delta) \wedge t(\overline{x}) = w] \cdot \Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[t(\overline{x}) = w \mid \overline{x} \in \mathrm{Low}(\delta)] \\
&\leq \sum_{w \in T} \delta_2 \cdot \Pr_{\overline{x} \leftarrow \overline{\mathbf{x}}}[t(\overline{x}) = w \mid \overline{x} \in \mathrm{Low}(\delta)] \leq \delta_2 \, .
\end{aligned}$$

□

Claim 4.10. *Let $x \in \mathrm{High}(\delta)$, then we have*

$$\Pr[\overline{\mathbf{z}} = \overline{y}] \geq (1-(1-\delta)^k)^n \cdot \frac{f(\overline{y})}{\mu} \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \,.$$

Proof. Consider $\mathrm{E}_{k,y_{\leq i}}$ to be the event that RejSam_k outputs one of its first k samples, when performed on $y_{\leq i}$. Then, it holds that

$$\Pr[\mathrm{E}_{k,y_{\leq i}}] = 1-(1-\hat{f}(y_{\leq i}))^k \geq 1-(1-\delta)^k \,.$$

On the other hand, we know that $\Pr[\overline{\mathbf{z}}_{i+1} = y_{i+1} \mid y_{\leq i} \wedge \mathrm{E}_{k,y_{\leq i}}] = \Pr[\overline{\mathbf{y}}_{i+1} = y_{i+1} \mid y_{\leq i}]$. Thus, we have

$$\begin{aligned}
\Pr[\overline{\mathbf{z}}_{i+1} = y_{i+1} \mid y_{\leq i}] &\geq \Pr[\overline{\mathbf{z}}_{i+1} = y_{i+1} \mid y_{\leq i} \wedge \mathrm{E}_{k,y_{\leq i}}] \cdot \Pr[\mathrm{E}_{k,y_{\leq i}}] \\
&= \Pr[\overline{\mathbf{y}}_{i+1} = y_{i+1} \mid y_{\leq i}] \cdot \Pr[\mathrm{E}_{k,y_{\leq i}}] \\
&\geq \Pr[\overline{\mathbf{y}}_{i+1} = y_{i+1} \mid y_{\leq i}] \cdot (1-(1-\delta)^k)^n.
\end{aligned}$$

By multiplying these inequalities for $i \in [n]$ we get $\Pr[\overline{\mathbf{z}} = \overline{y}] \geq (1-(1-\delta)^k)^n \cdot \Pr[\overline{\mathbf{y}} = \overline{x}]$. □

Claim 4.11. *For $\delta_1 \cdot \delta_2 = \delta$, it holds that*

$$\mu_k \geq \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) - \frac{\delta_1+\delta_2}{\mu} - n \cdot (1-\delta)^k \,.$$

Proof. Let

$$\mu' = \sum_{\overline{y} \in \mathrm{Low}(\delta) \cap \mathrm{Small}(\delta_1)} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \,,$$

$$\text{and} \quad \mu'' = \sum_{\overline{y} \in \mathrm{Low}(\delta) \cap \mathrm{Big}(\delta_1)} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \,.$$

By Claim, 4.5 we have

$$\begin{aligned}
\mathbb{E}[f(\overline{\mathbf{z}})] &\geq \sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{\Pr[\overline{\mathbf{z}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\
&\geq \sum_{\overline{y} \in \mathrm{High}(\delta)} \left(\frac{\Pr[\overline{\mathbf{z}}^{[n]} = \overline{y}]}{\Pr[\overline{\mathbf{x}} = \overline{y}]}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\
\text{(by Claim 4.10)} \quad &\geq \sum_{\overline{y} \in \mathrm{High}(\delta)} (1-(1-\delta)^k)^{n \cdot p} \cdot \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) \\
&= (1-(1-\delta)^k)^{n \cdot p} \cdot \left(\sum_{\overline{y} \in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \cdot f(\overline{y}) - \mu' - \mu'' \right).
\end{aligned}$$

We have $\mu' \leq \delta_1^{1+p}/\mu^p \leq \delta_1/\mu$, because $f(\overline{y}) \leq \delta_1$ for all $\overline{y} \in \mathrm{Small}(\delta_1)$. Also, by Claim 4.9, we get

$$\mu'' \leq \sum_{\overline{y} \in \mathrm{Low}(\delta) \cap \mathrm{Big}(\delta_1)} \left(\frac{1}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}} = \overline{y}] \leq \frac{\delta_2}{\mu^p} \leq \frac{\delta_2}{\mu} \,.$$

Therefore, we have

$$\mathbb{E}[f(\overline{\mathbf{z}})] \geq (1-(1-\delta)^k)^{n\cdot p} \cdot \left(\sum_{\overline{y}\in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}}=\overline{y}] \cdot f(\overline{y}) - \frac{\delta_1+\delta_2}{\mu} \right)$$

$$\text{(by Bernoulli inequality)} \quad \geq (1-n\cdot(1-\delta)^k) \cdot \left(\sum_{\overline{y}\in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}}=\overline{y}] \cdot f(\overline{y}) - \frac{\delta_1+\delta_2}{\mu} \right)$$

$$\geq \sum_{\overline{y}\in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}}=\overline{y}] \cdot f(\overline{y}) - \frac{\delta_1+\delta_2}{\mu} - n\cdot(1-\delta)^k.$$

□

In order to conclude the proof of Claim 4.7, we can set $\delta_1 = \delta_2 = \sqrt{\delta}$ and let $\delta \leq (\varepsilon\mu/4)^2$. Then, given that we have $k \geq \frac{16\ln(2n/\varepsilon)}{\varepsilon^2\mu^2}$, we get

$$\mathbb{E}[f(\overline{\mathbf{z}})] \geq \sum_{\overline{y}\in \mathrm{Supp}(\overline{\mathbf{x}})} \left(\frac{f(\overline{y})}{\mu}\right)^p \cdot \Pr[\overline{\mathbf{x}}=\overline{y}] \cdot f(\overline{y}) - \frac{\varepsilon}{2} - \frac{\varepsilon}{2}\,.$$

□

4.4 Relating the Bias to the Variance: Proving Lemma 3.7

We use Lemma A.2 by letting $\varphi(x) = x^{1+p}$. Thus, we have to minimize the following function on $x \in [0,1]$,

$$g_\mu(x) = \left(x^{1+p} - \mu^{1+p} - (1+p)\cdot\mu^p\cdot(x-\mu)\right)/(x-\mu)^2\,.$$

We now prove that the minimum happens on $x = 1$. Note that the function $g_\mu(x)$ is continues on $[0,\mu)$ and $(\mu,0]$ and the limit exists at $x=\mu$ and is equal to $1/2\cdot p\cdot(1+p)\cdot\mu^{-1+p}$. Therefore if we show that g'_μ is negative for $x \in [0,\mu)\cup(\mu,1]$ it implies that, $\forall x\in[0,1] g(x) \geq g(1)$. We have

$$g'_\mu(x) = \frac{(p-1)\cdot\mu^{p+1} - (p+1)\cdot x\cdot\mu^p + (p+1)\cdot\mu\cdot x^p - (p-1)\cdot x^{p+1}}{(\mu-x)^3}$$

$$\text{(using } c = x/\mu) \quad = \mu^{p-2}\cdot\frac{(p-1)-(p+1)\cdot c + (p+1)\cdot c^p - (p-1)\cdot c^{p+1}}{(1-c)^3}\,.$$

We prove that the numerator $q(c) = (p-1)-(p+1)\cdot c+(p+1)\cdot c^p-(p-1)\cdot c^{p+1}$ is positive for $c>1$ and negative for $0<c<1$. For $c>0$, we have

$$\begin{aligned} q'(c) &= -(1+p) + (p+1)\cdot p\cdot c^{p-1} + (1-p)\cdot(p+1)\cdot c^p \\ &= (1+p)\cdot(p\cdot c^{p-1} + (1-p)\cdot c^p - 1) \\ \text{(by AM-GM inequality of Lemma A.1)} \quad &\geq (1+p)\cdot(c^{p\cdot(p-1)}\cdot c^{(1-p)\cdot p} - 1) \\ &= 0\,. \end{aligned}$$

Therefore, q is increasing for $c>0$ which implies $\forall c\in[0,1], q(c) < q(1) = 0$ and $\forall c>1, q(c) > q(1) = 0$. We have $\forall x\in[0,\mu)\cup(\mu,1], g'(x)\leq 0$. Therefore we have

$$\forall x\in[0,1], g_u(x) \geq g_u(1)\,. \tag{3}$$

Now we prove that $g_\mu(1) \geq \frac{p(1+p)}{2}$. Consider the following function,

$$w(\mu) = g_\mu(1) = \left(1 - \mu^{1+p} - (1+p) \cdot \mu^p \cdot (1-\mu)\right) / (1-\mu)^2 \,.$$

We will show that q is a decreasing function for $\mu \in [0,1]$. We have

$$w'(\mu) = \frac{p \cdot (1-\mu^2) \cdot \mu^{p-1} + p^2(1-\mu)^2 \cdot \mu^{p-1} + 2 \cdot (\mu^p - 1))}{(-1+\mu)^3} \,.$$

We will show that the numerator $s(\mu) = p \cdot (1-\mu^2) \cdot \mu^{p-1} + p^2(1-\mu)^2 \cdot \mu^{p-1} + 2 \cdot (\mu^p - 1)$ is negative for $\mu \in [0,1]$. We have $s'(\mu) = p(p^2-1) \cdot (1-\mu)^2 \cdot \mu^{p-2}$ which is negative for $\mu \in [0,1]$. This implies that $\forall \mu \in [0,1], s(\mu) \geq s(1) = 0$. Therefore, w is a decreasing function, and we obtain

$$\forall \mu \in [0,1], g_\mu(1) = w(\mu) \geq \lim_{u \to 1} w(u) = \frac{p(1+p)}{2} \,. \tag{4}$$

Now, we conclude that

$$\begin{aligned}
\mu^{-p} \cdot \mathbb{E}[\boldsymbol{\alpha}^{1+p}] - \mu &= \mu^{-p} \left(\mathbb{E}[\boldsymbol{\alpha}^{1+p}] - \mu^{1+p}\right) \\
\text{(by Lemma A.2)} \quad &\geq \mu^{-p} \left(\inf_{x \in [0,1]} \{g_\mu(x)\} \cdot \nu \right) \\
\text{(by Inequality 3)} \quad &\geq \mu^{-p} \cdot g_\mu(1) \cdot \nu \\
\text{(by Inequality 4)} \quad &\geq \frac{p \cdot (1+p)}{2 \cdot \mu^p} \cdot \nu \,.
\end{aligned}$$

A Some Useful Inequalities

The following well-known variant of the inequality for the arithmetic mean and the geometric mean could be derived from the Jensen's inequality.

Lemma A.1 (Weighted AM-GM inequality)**.** *For any $n \in \mathbb{N}$, let $z_1, ..., z_n$ be a sequence of non-negative real numbers and let $w_1, ..., w_n$ be such that $w_i \geq 0$ for every $i \in [n]$ and $\sum_{i=1}^n w_i = 1$. Then, it holds that*

$$\sum_{i=1}^n w_i z_i \geq \prod_{i=1}^n z_i^{w_i} \,.$$

The following lemma provides a tool for lower bounding the gap between the two sides of Jensen's inequality, also known as the Jensen gap.

Lemma A.2 (Lower bound for Jensen gap [LB17])**.** *Let $\boldsymbol{\alpha}$ be a real-valued random variable,* $\mathrm{Supp}(\boldsymbol{\alpha}) \subseteq [0,1]$*, and $\mathbb{E}[\boldsymbol{\alpha}] = \mu$. Let $\varphi(\cdot)$ be twice differentiable on $[0,1]$, and let $h_b(a) = \frac{\varphi(a)-\varphi(b)}{(a-b)^2} - \frac{\varphi'(a)}{a-b}$. Then,*

$$\mathbb{E}[\varphi(\boldsymbol{\alpha})] - \varphi(\mu) \geq \mathbb{V}[\boldsymbol{\alpha}] \cdot \inf_{a \in [0,1]} \{h_\mu(a)\} \,.$$

References

[ABL14] 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. 2

[ACM+14] 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, 12

[ACM+17] Per Austrin, Kai-Min Chung, Mohammad Mahmoody, Rafael Pass, and Karn Seth. On the impossibility of cryptography with tamperable randomness. *Algorithmica*, 79(4):1052–1101, Dec 2017. 1, 3, 5

[BBEG18] Salman Beigi, Andrej Bogdanov, Omid Etesami, and Siyao Guo. Optimal deterministic extractors for generalized santha-vazirani sources. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 116. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. 5

[BEG17] 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. 5

[BEK02] Nader H. Bshouty, Nadav Eiron, and Eyal Kushilevitz. PAC learning with nasty noise. *Theoretical Computer Science*, 288(2):255–275, 2002. 2

[BGS+17] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pages 119–129, 2017. 3

[BHLT17] Niv Buchbinder, Iftach Haitner, Nissan Levi, and Eliad Tsfadia. Fair coin flipping: Tighter analysis and the many-party case. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2580–2600. Society for Industrial and Applied Mathematics, 2017. 5

[BHT14] Itay Berman, Iftach Haitner, and Aris Tentes. Coin flipping of any constant bias implies one-way functions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 398–407. ACM, 2014. 5

[BHT18] Itay Berman, Iftach Haitner, and Aris Tentes. Coin flipping of any constant bias implies one-way functions. *Journal of the ACM (JACM)*, 65(3):14, 2018. 5

[BIK+17] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191. ACM, 2017. 3

[BOL89] M. Ben-Or and N. Linial. Collective coin flipping. *Randomness and Computation*, 5:91–115, 1989. 1, 3, 6, 7, 12

[BVH+18] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018. 3

[CFGN96] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 639–648. ACM, 1996. 9

[CG88] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988. 5

[CI93] Richard Cleve and Russell Impagliazzo. Martingales, collective coin flipping and discrete control processes. *In other words*, 1:5, 1993. 5

[Dod01] Yevgeniy Dodis. New imperfect random source with applications to coin-flipping. In *International Colloquium on Automata, Languages, and Programming*, pages 297–309. Springer, 2001. 5, 6

[DOPS04] 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. 5

[DR06] Yevgeniy Dodis and Renato Renner. On the impossibility of extracting classical randomness using a quantum computer. In *International Colloquium on Automata, Languages, and Programming*, pages 204–215. Springer, 2006. 5

[DY15] Yevgeniy Dodis and Yanqing Yao. Privacy with imperfect randomness. In *Annual Cryptology Conference*, pages 463–482. Springer, 2015. 5

[FYB18] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018. 3

[GKP15] 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. 5

[HO14] Iftach Haitner and Eran Omri. Coin flipping with constant bias implies one-way functions. *SIAM Journal on Computing*, 43(2):389–409, 2014. 1, 3, 5, 6, 7, 12

[KL93] Michael J. Kearns and Ming Li. Learning in the Presence of Malicious Errors. *SIAM Journal on Computing*, 22(4):807–837, 1993. 2

[KMY+16] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016. 3

[LB17] JG Liao and Arthur Berg. Sharpening Jensen's inequality. *The American Statistician*, (accepted in 2017). 7, 19

[LLS89] David Lichtenstein, Nathan Linial, and Michael Saks. Some extremal problems arising from discrete control processes. *Combinatorica*, 9(3):269–287, 1989. 5

[MDM18] Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. Learning under p-tampering attacks. In *Algorithmic Learning Theory*, pages 572–596, 2018. 1, 2, 3, 12

[MM17] 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. 1, 2, 3, 5, 6, 12

[MMR+16] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016. 3

[MPS10] Hemanta K Maji, Manoj Prabhakaran, and Amit Sahai. On the computational complexity of coin flipping. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 613–622. IEEE, 2010. 5

[MR17] Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. *Google Research Blog*, 2017. 3

[PMSW16] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814*, 2016. 2

[RVW04] Omer Reingold, Salil Vadhan, and Avi Wigderson. A note on extracting randomness from santha-vazirani sources. *Unpublished manuscript*, 2004. 5

[STS16] 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. 2

[SV86] Miklos Santha and Umesh V. Vazirani. Generating quasi-random sequences from semi-random sources. *J. Comput. Syst. Sci.*, 33(1):75–87, 1986. 5

[Val85] Leslie G. Valiant. Learning disjunctions of conjunctions. In *IJCAI*, pages 560–566, 1985. 2

[XBB+15] 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. 2