UNIVERSAL MULTI-PARTY POISONING ATTACKS

Saeed Mahloujifar[*], **Mohammad Mahmoody**[†], **Ameer Mohammed**[‡]

ABSTRACT

In a poisoning attack, an adversary tampers with a fraction of the training data $\mathcal{T}$. In the distributed multi-party setting, $\mathcal{T}$ might be gathered gradually from m data providers $P_1, \dots, P_m$ who submit their shares of $\mathcal{T}$ interactively. A multi-party poisoning adversary might control k of these m parties to hurt the learning.

In this work, we demonstrate universal multi-party poisoning attacks that adapt and apply to any multi-party learning process with arbitrary interaction pattern between the parties. More generally, we introduce and study (k, p)-poisoning attacks in which an adversary controls $k \in [m]$ of the parties, and 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 becomes the traditional notion of single-party poisoning, and for $p = 1$ it coincides with the standard notion of static corruption in secure multi-party computation in cryptography.

We prove that for any "bad" property B of the final trained hypothesis h (e.g., h failing on a particular test example or having "large" risk) that has an arbitrarily small constant probability of happening without the attack, there always is a (k, p)-poisoning attack that increases the probability of B by $\Omega(p \cdot k/m)$. Our provable attacks achieve this by only using clean labels. At a technical level, we prove a general result about biasing the average output of any bounded function $f(x_1, \dots, x_n) \in [0, 1]$ through a polynomial-time online attack in which each input x_i might be controlled by an adversary (who only knows the previously sampled inputs) with *marginal* probability p. Previous works have studied *independent* probabilities, preventing the application to multiparty corruption.

Though our adversary only controls the actual messages of $k < m$ parties, it needs fresh (new samples) from the data distribution of uncorrupted parties as well. Extending our attacks to not rely on this feature, or finding defenses in such settings remains as an interesting open question.

1 INTRODUCTION

Learning from a set $\mathcal{T} = \{d_1 = (a_1, b_1), \dots, d_n = (a_n, b_n)\}$ 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 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 (e.g., due to noise in the data).

Poisoning attacks. Many previous works studying noise in the data allow it to be *adversarial* and *maliciously* chosen against the learner (Valiant, 1985; Kearns & Li, 1993; Bshouty et al., 2002). A tightly related and more recent approach to the problem of learning under adversarial noise is the framework of so-called *poisoning* (aka *causative*) attacks (Barreno et al., 2006; Biggio et al., 2012; Papernot et al., 2016), 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 or make it more likely to fail on a particular test instance (i.e., getting a *targeted* poisoning attack (Barreno et al., 2006; Shen et al., 2016)).

[*]University of Virginia (`saeed@virginia.edu`).

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

[‡]University of Kuwait (`ameer.mohammed@ku.edu.kw`).

Multi-party poisoning. In the distributed setting (McMahan & Ramage, 2017; McMahan et al., 2016; Bonawitz et al., 2017; Konečnỳ et al., 2016), the training data $\mathcal{T}$ might be coming from various sources; e.g., it can be generated by m data providers $P_1, \dots, P_m$ 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 $P_1, \dots, P_m$ is to eventually help G construct a hypothesis h that does well (e.g. in the case of classification) 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 $\mathbf{D}$. To model this more general setting, 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 (Fung et al., 2018; Bagdasaryan et al., 2018; Blanchard et al., 2017)) 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?

Answering the above question is critical to understand the *limits* of provable security against poisoning attacks in the multi-party setting.

1.1 Our Contribution

We first formalize a new general notion of multi-party poisoning. We then develop attacks for cryptographic coin tossing protocols that imply data poisoning attacks in the multi-party setting.

New attack model: (k, p)-poisoning attacks. our first contribution of this work is to formalize a general notion that covers multi-party poisoning attackers that corrupt k out of m data provider parties and furthermore, for each message sent by a corrupted party, the adversary still generates data that is "close" to the honestly generated data. More formally, a (k, p)-poisoning attacker Adv can first choose to corrupt k of the parties. Then, 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 p to the original distribution $\mathbf{D}_i$ in total variation distance. Our (k, p)-poisoning attacks include the so called "p-tampering" attacks of (Mahloujifar et al., 2018a) as special case by letting $k = m$ (m is the number of parties). Moreover, (k, p)-attacks also include the standard model of k static corruption in secure multi-party computation (in cryptography) letting $p = 1$. Our main result in this works is to prove the *universal* power of (k, p)-poisoning as follow. We show that in *any* m-party learning protocol in which the produced hypothesis h has a bad property B (e.g., failing on a particular target instance known to the adversary) with probability μ, then this probability can be increased to $\mu' = \mu^{1-kp/m}$. For example, by corrupting half of the parties (i.e., $p = 1, k = m/2$) the adversary can increase the probability of B from $1/100$ to $1/10$.

Universal nature of our attack. Our attacks are *universal* in the sense that they could be applied to *any* learning algorithm for *any* learning task, and they are *dimension-independent* as they applied to any data distribution. On the other hand, our universal attacks rely on an initial vulnerability of arbitrary small *constant* probability that is then amplified through the poisoning attack. As a result, although recent poisoning attacks (e.g., see (Koh et al., 2018)) obtain *stronger* bounds in their attack against specific defenses, our attacks apply to *any* algorithm with any built in defenses.

Deriving attacks on federated learning as special case. Since we allow the distribution of each party in the multi-party case to be completely dependent on that party, our attacks cover the case of model poisoning in federated learning (Bagdasaryan et al., 2018; Bhagoji et al., 2018), in which each party sends something *other* than their plain share of data, as special case. In fact, multiple works have already demonstrated the power of poisoning attacks in the federated learning setting (e.g., see (Fung et al., 2018; Bhagoji et al., 2018; Chen et al., 2018; 2017; Guerraoui et al., 2018; Yin et al., 2018)). Some of these attacks obtain stronger quantitative bounds in their attacks, however this is anticipated as these works investigate attacks on *specific* learners, while a crucial property of our attack is that our attacks come with *provable* bounds and are *universal* in that they apply to *any* learning task and *any* hypothesis class (including neural nets as special casse), so long as there is an initial $\Omega(1)$ vulnerability (for some bad property) over the generated hypothesis.

Note that, our attacks actually do not need the exact history of examples that are used by parties, and only need to know the updates sent by the parties during the course of protocol. Suppose an uncorrected party randomizes its local model (e.g., for differential privacy purposes) and shares an update u_i with the server. Knowledge of u_i is enough for our attacker. One might go even further and ask what if the updates are sent in a secure/private way? Interestingly, our attack work in that model too as it only needs to know the effect of the updates on the central model at the end of round $i-1$ (because all attack wants is to perform a random continuation on the intermediate model).

It also worth mentioning that our attack requires sampling oracles from distributions of all the parties. This might seem that we are giving the adversary too much power. However, we think the right way to define security of federated learning is by giving the adversary everything that hat might be leaked to them. This way of defining security is inspired by cryptography. For instance, when modeling the chosen plaintext security of encryption schemes, adversary is given access to an encryption oracle, while one might question how realistic it is. Analogously, In federated learning, the adversary can potentially gather some statistics about the distribution of other parties and learn them over time. However, as mentioned above, we do not need to give adversary access to the actual data of honest parties. Only the public effect of them on the shared model is needed.

Related work. Recent breakthroughs of Diakonikolas et al. (2016) and Lai et al. (Lai et al., 2016) demonstrated the surprising power of robust distribution learning over poisoned training samples with limited risk that does not depend on the dimension of the distribution (but still depends on the fraction of poisoned data). These works led to an active line of work (e.g., see (Charikar et al., 2017; Diakonikolas et al., 2017; Balakrishnan et al., 2017; Diakonikolas et al., 2018b;a; Prasad et al., 2018; Diakonikolas et al., 2018c;a; Prasad et al., 2018; Charikar et al., 2017; Diakonikolas et al., 2018b) and references therein). On the negative side, Mahloujifar, Mahmoody and Diochnos (Mahloujifar & Mahmoody, 2017; Mahloujifar et al., 2018b) studied (universal) poisoning attacks on single party setting. These attacks were also universal. Similarly, these attacks also rely on an initial vulnerability of arbitrary small *constant* probability that is then amplified through the poisoning attack. That is why such universal attacks do not contradict the positive results mentioned above.

2 Multi-Party Poisoning: Definitions and the Main Result

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 $\boldsymbol{\alpha}$. A learning problem $(\mathcal{A}, \mathcal{B}, \mathbf{D}, \mathcal{H})$ 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 $\mathcal{A} \times \mathcal{B}$.[1] The set $\mathcal{H} \subseteq \mathcal{B}^{\mathcal{A}}$ is called the *hypothesis space* or *hypothesis class*. An *example* s is a pair $s = (a, b)$ where $x \in \mathcal{A}$ and $y \in \mathcal{B}$.

Definition 2.1 (Multi-party learning protocols)**.** *An m-party learning protocol Π for the m-party learning problem $(\mathcal{D}, \mathcal{H})$ consists of an aggregator function G and m (interactive) data providers $\mathcal{P} = \{P_1, \dots, P_m\}$. 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 $(a, b) \leftarrow \mathbf{D}_i$.[2] In the last round, the aggregator function G maps the the messages to an output hypothesis $h \in \mathcal{H}$.*

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 control this sample using the tampered distribution $\widetilde{\mathbf{D}}$ such that*

[1]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).

[2]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.

$|\widetilde{\mathbf{D}} - \mathbf{D}_i| \leq p$ in total variation 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.* Adv *is 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).*

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

Theorem 2.3 (Power of efficient multi-party poisoning)**.** *For an m-party protocol Π for parties $\mathcal{P} = \{P_1, \dots, P_m\}$, let B be any (polynomial-time testable) predicate that happens over the learned hypothesis with probability μ when no attack happens. Then and any $p \in [0, 1]$ and $k \in [m]$, there is a* plausible *(i.e., clean-label), (k, p)-poisoning attack* Adv *that runs in time* $\mathrm{poly}(m/\varepsilon)$ *and increases the probability of the bad predicate B from μ to $\mu' = \mu^{1-kp/m} - \varepsilon$.*

Before proving Theorem 2.3, we need to develop our main result about the power of generalized p-tampering attacks. In Section 3, we develop such tools and show how to prove Theorem 2.3.

3 Multi-Party Poisoning through Generalized p-Tampering

To prove our Theorem 2.3 we interpret the multi-party learning protocol as a coin tossing protocol in which the final bit is 1 if h has the (bad) property B. We define a corresponding attack model in coin tossing protocols that can be directly used to obtain the desired goal; this model is called *generalized* p-tampering. Below, we formally state our main result about the power of generalized p-tampering attacks. we start by formalizing some notation and basic definitions.

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 $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ we refer to a joint distribution over vectors with n components. For a joint distribution $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \dots, \mathbf{x}_n)$, we use $\mathbf{x}_{\leq i}$ to denote the joint distribution of the first i variables $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \dots, \mathbf{x}_i)$. Also, for a vector $\overline{x} = (x_1 \dots x_n)$ 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 $\mathbf{D}$. 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 valid prefixes of $\overline{\mathbf{x}}$.

Definition 3.1 (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 the maximum bit length of any $\overline{x} \in \mathrm{Supp}(\overline{\mathbf{x}})$.*

Definition 3.2 (Online samplers)**.** *We call OnSam an* online sampler *for $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \dots, \mathbf{x}_n)$ if for all $x_{\leq i-1} \in \mathrm{ValPref}(\overline{\mathbf{x}})$, $\mathsf{OnSam}(n, x_{\leq i-1}) \equiv \mathbf{x}_i$. 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 $\overline{x} \in \mathrm{Supp}(\overline{\mathbf{x}})$.*

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 $\overline{y} \leftarrow \langle \overline{\mathbf{x}} \parallel \mathsf{T}, S \rangle$.

Definition 3.3 (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 $i \in [n], \Pr_{S \leftarrow \mathbf{S}}[i \in S] = p$. (Note that $i \in \mathbf{S}$ and $j \in \mathbf{S}$ could be correlated events.)*

The following theorem states the power of generalized p-tampering attacks.

Theorem 3.4 (Biasing through generalizing p-tampering)**.** *Let $\mathbf{S}$ be a p-covering distribution on $[n]$, $\overline{\mathbf{x}} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a joint distribution, $f\colon \operatorname{Supp}(\overline{\mathbf{x}}) \mapsto [0,1]$, and $\mu = \mathbb{E}[f(\overline{\mathbf{x}})]$. Then, 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 $\operatorname{poly}(N/\varepsilon)$, 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 (Ben-Or & Linial, 1989) 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 (Haitner & Omri, 2014) with extra properties). Even for this case, compared to (Ben-Or & Linial, 1989; Haitner & Omri, 2014) 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 (Haitner & Omri, 2014) 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 (Ben-Or & Linial, 1989; Haitner & Omri, 2014) 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.

Proving Theorem 3.4. The construction below describes the biasing algorithm of our Theorem 3.4.

Construction 3.5 (k-rejection-sampling tampering)**.** *Let $\overline{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a joint distribution and $f\colon \operatorname{Supp}(\overline{\mathbf{x}}) \mapsto [0,1]$. The k-*rejection sampling *tampering algorithm* RejSam_k^f *works as follows. Given the $y_{\leq i-1} \in \operatorname{ValPref}(\overline{\mathbf{x}})$, the tampering algorithm would do the following 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 k iterations, 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 $k \to \infty$.

In the full version of this paper, we prove the following claim which proves Theorem 2.3.

Claim 3.6. *Let $\overline{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a joint distribution and $f\colon \operatorname{Supp}(\overline{\mathbf{x}}) \mapsto [0,1]$. For any $\varepsilon \in [0,1]$, let $k \geq \frac{16\ln(2n/\varepsilon)}{\varepsilon^2\mu^2}$. Then RejSam_k runs in time $O(k) = \operatorname{poly}(N/(\varepsilon \cdot \mu))$, 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*

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

Sketch of deriving Theorem 2.3 from Theorem 3.4. 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 use this covering to prove the theorem. 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 $\overline{\mathbf{x}} = \mathbf{D}_{w(1)} \times \cdots \times \mathbf{D}_{w(n)}$.

We define a function $f : \operatorname{Supp}(\overline{\mathbf{x}}) \to \{0,1\}$, which is a Boolean function and is 1 if the output of the protocol has the property B, and otherwise it is 0. Now we use Theorem 3.4. We know that $\mathbf{S}$ is a $(p \cdot \frac{k}{m})$-covering for $[n]$. Therefore of Theorem 3.4, there exist an $\operatorname{poly}(m/\varepsilon)$ time tampering algorithm T_ε that changes $\overline{\mathbf{x}}$ to $\overline{\mathbf{y}} \equiv \langle \overline{\mathbf{x}} \parallel \mathsf{T}_\varepsilon^{f,\mathsf{OnSam}}, \mathbf{S}\rangle$ where $\mathbb{E}[f(\overline{\mathbf{y}})] \geq \mathbb{E}[f(\overline{\mathbf{y}})]^{1-pk/m} - \varepsilon$.

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 average output at least $\mathbb{E}[f(\overline{\mathbf{y}})]^{1-pk/m} - \varepsilon$. 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 $|\overline{\mathbf{x}} - \langle \overline{\mathbf{x}} \parallel \mathsf{T}_\varepsilon^{f,\mathsf{OnSam}}, \mathbf{Bion}(R_C, p)\rangle|$ is at most p. This concludes the proof. □

REFERENCES

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

Sivaraman Balakrishnan, Simon S Du, Jerry Li, and Aarti Singh. Computationally efficient robust sparse estimation in high dimensions. In *Conference on Learning Theory*, pp. 169–212, 2017.

Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pp. 16–25. ACM, 2006.

M. Ben-Or and N. Linial. Collective coin flipping. *Randomness and Computation*, 5:91–115, 1989.

Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. *arXiv preprint arXiv:1811.12470*, 2018.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pp. 1467–1474. Omnipress, 2012.

Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pp. 119–129, 2017.

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*, pp. 1175–1191. ACM, 2017.

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

Moses Charikar, Jacob Steinhardt, and Gregory Valiant. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 47–60. ACM, 2017.

Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Draco: Byzantine-resilient distributed training via redundant gradients. In *International Conference on Machine Learning*, pp. 902–911, 2018.

Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):44, 2017.

Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high dimensions without the computational intractability. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pp. 655–664. IEEE, 2016.

Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. Statistical query lower bounds for robust estimation of high-dimensional Gaussians and Gaussian mixtures. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pp. 73–84. IEEE, 2017.

Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. *arXiv preprint arXiv:1803.02815*, 2018a.

Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. List-decodable robust mean estimation and learning mixtures of spherical Gaussians. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1047–1060. ACM, 2018b.

Ilias Diakonikolas, Weihao Kong, and Alistair Stewart. Efficient algorithms and lower bounds for robust linear regression. *arXiv preprint arXiv:1806.00040*, 2018c.

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

Rachid Guerraoui, Sébastien Rouault, et al. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, pp. 3518–3527, 2018.

Iftach Haitner and Eran Omri. Coin flipping with constant bias implies one-way functions. *SIAM Journal on Computing*, 43(2):389–409, 2014.

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

Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. *arXiv preprint arXiv:1811.00741*, 2018.

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.

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

Saeed Mahloujifar and Mohammad Mahmoody. Blockwise p-Tampering Attacks on Cryptographic Primitives, Extractors, and Learners. In *Theory of Cryptography Conference*, pp. 245–279. Springer, 2017.

Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. Learning under p-Tampering Attacks. In *ALT*, pp. 572–596, 2018a.

Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. Learning under p-tampering attacks. In *Algorithmic Learning Theory*, pp. 572–596, 2018b.

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

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.

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.

Adarsh Prasad, Arun Sai Suggala, Sivaraman Balakrishnan, and Pradeep Ravikumar. Robust estimation via robust gradient estimation. *arXiv preprint arXiv:1802.06485*, 2018.

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*, pp. 508–519. ACM, 2016.

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

Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018.