Blockwise p-Tampering Attacks on Cryptographic Primitives, Extractors, and Learners

Saeed Mahloujifar(✉) and Mohammad Mahmoody

University of Virginia, Charlottesville, USA
{saeed,mohammad}@virginia.edu

Abstract. Austrin et al. [1] studied the notion of bitwise p-tampering attacks over randomized algorithms in which an efficient 'virus' gets to control each bit of the randomness with independent probability p in an online way. The work of [1] showed how to break certain 'privacy primitives' (e.g., encryption, commitments, etc.) through bitwise p-tampering, by giving a bitwise p-tampering *biasing* attack for increasing the average $\mathbb{E}[f(U_n)]$ of any efficient function $f\colon \{0,1\}^n \mapsto [-1,+1]$ by $\Omega(p \cdot \mathrm{Var}[f(U_n)])$.

In this work, we revisit and extend the bitwise tampering model of [1] to *blockwise* setting, where blocks of randomness becomes tamperable with independent probability p. Our main result is an efficient blockwise p-tampering attack to bias the average $\mathbb{E}[f(\overline{X})]$ of any efficient function f mapping arbitrary $\overline{X}$ to $[-1,+1]$ by $\Omega(p \cdot \mathrm{Var}[f(\overline{X})])$ *regardless* of how $\overline{X}$ is partitioned into individually tamperable blocks $\overline{X} = (X_1, \ldots, X_n)$. Relying on previous works of [1,19,36], our main biasing attack immediately implies efficient attacks against the privacy primitives as well as seedless multi-source extractors, in a model where the attacker gets to tamper with each block (or source) of the randomness with independent probability p. Further, we show how to increase the classification error of deterministic learners in the so called 'targeted poisoning' attack model under Valiant's adversarial noise. In this model, an attacker has a 'target' test data d in mind and wishes to increase the error of classifying d while she gets to tamper with each training example with independent probability p an in an online way.

1 Introduction

In this work, we study *tampering* attacks that efficiently manipulate the *randomness* of randomized algorithms with adversarial goals in mind. Tampering attacks could naturally be studied in the context of cryptographic algorithms that (wish to) access perfectly uniform and untampered randomness for sake of

S. Mahloujifar—Supported by University of Virginia's SEAS Research Innovation Award.

M. Mahmoody—Supported by NSF CAREER award CCF-1350939, and University of Virginia's SEAS Research Innovation Award.

Y. Kalai and L. Reyzin (Eds.): TCC 2017, Part II, LNCS 10678, pp. 245–279, 2017.
https://doi.org/10.1007/978-3-319-70503-3_8

achieving security. However, the scope of such attacks goes beyond the context of cryptography and could be studied more broadly for any class of algorithms that depend on some form of untampered random input and try to achieve specific goals (e.g., learning algorithms using untampered training data to generate a hypothesis). Here, we are interested in understanding the power and limitations of such tampering attacks over the randomness, when the adversary can tamper with, or even control, $\approx p$ fraction of the randomness.[1]

The most relevant to our study here is the work of Austrin et al. [1] that introduced the notion of *bitwise p-tampering* attacks on the randomness of cryptographic primitives. In this model, the adversary generates an efficient 'virus' who gets into the 'infected' device, can read everything, but is limited in what it can alter. As the stream of bits of randomness $R = (r_1, \ldots, r_n)$ is being generated, for every bit r_i, the p-tampering virus gets to change r_i with independent probability p (i.e., with probability $(1-p)$ the bit remains unchanged). p-tampering attacks are online, so the virus does not know the future incoming bits, but it can base its decisions based on the history of the (potentially tampered) bits. The work of [1] proved that bitwise p-tampering attacks can always increase the average of efficient bounded functions $f\colon \{0,1\}^n \mapsto [-1,+1]$ by $\Omega(p \cdot \mathrm{Var}[f(U_n)])$ where $\mathrm{Var}[f(U_n)]$ is the variance of $f(U_n)$.

Austrin et al. [1] showed how to break a variety of 'privacy' cryptographic primitives (e.g., public-key and private key encryption, zero knowledge, commitments, etc.) that have 'indistinguishability-based' security games using their main efficient bitwise p-tampering biasing attack. In such cryptographic attacks, the *code* of the p-tampering virus is generated by an outside adversary who only knows the public information (e.g. public key). Previously, Dodis et al. [19] had shown that for the same cryptographic primitives, there are high-min-entropy Santha-Vazirani sources of randomness [39] that make them insecure. Thus, the work of [1] was a strengthening of the results of [19] showing how to generate such 'bad' SV sources through efficient p-tampering attacks. The p-tampering attacks of [1], and in particular their core attack for biasing the output of balanced bounded functions, crucially depend on the fact that the attacker can tamper with *every single bit* of the randomness *independently* with probability p. However, randomness is usually generated in blocks rather than bits [4,16,21,28], e.g., during the boot time [30], and is also made available to the algorithms requesting them in blocks. Thus, it is indeed natural to consider tampering attackers who sometimes get to change an incoming *block* of randomness.

Blockwise p-tampering attacks. In this work, we revisit the bitwise p-tampering model of [1] and extend it to a setting where the tampering could happen over blocks. Suppose A is an algorithm taking $\overline{X} = (X_1 \times \cdots \times X_n)$ as input where $\overline{X}$ is a distribution consisting of n blocks and the i'th block is independently sampled from the distribution X_i. For example, A could be a cryptograhpic algorithm in which X_i is the i'th block of *uniform* randomness given to A. Or A could also be a learning algorithm given n i.i.d training examples. Roughly speaking, a

[1] Note that if the adversary can control all the randomness, we are effectively back to what we can do in the deterministic setting.

blockwise p-tampering attack on (the randomness of) A is an algorithm Tam working as follows. Suppose we sample the blocks $x_i \leftarrow X_i$ one by one. Then the i'th block x_i becomes 'tamperable' with independent probability p for each i, and it remains intact with probability $1-p$. In case x_i becomes tamperable, then Tam could substitute x_i with another value x_i' in the support set[2] of X_i in an online way. Namely, when Tam gets the chance to tamper with x_i it could decide on a new block x_i' based on the knowledge of previous (tampered) blocks. The tampering algorithm Tam could also depend on (and thus know everything about) the algorithm A including all of its inputs selected so far, but it cannot write anything except when it is given the chance to tamper with a block of randomness.

Different p-tampering attackers could pursue different goals. For example, as it was done in the bitwise setting of [1], a p-tampering attack might aim to 'signal out' a secret information (e.g., the plain-text). Another example is when Tam wants to increase the classification error of the hypothesis output by a learner A where each block $x_i = (d, t)$ consists of a labeled example sampled from the same distribution.

We also note that, though called primarily a tampering attack, p-tampering attacks are not blind tampering attackers and naturally rely on the knowledge of the previous random bits before deciding on the tampering of the next bit/block, although such knowledge is only given to the tampering virus, and e.g., not the external adversary generating the code of the virus. That is a reason why the proven power of p-tampering attacks in this work is not in contradiction with known *positive* results such as [18,24,26,32] that guarantee tamper resilience.

1.1 Our Results

Our main result is a generalization of the biasing attack of [1] to the blockwise setting. We first describe this result, and then we will describe some of the applications of this biasing attack.

Theorem 1 (Informally stated). *Let $\overline{X} = (X_1 \times \cdots \times X_n)$ be a product distribution where each of X_i's is efficiently samplable. For any efficient function $f\colon \mathrm{Supp}(\overline{X}) \mapsto [-1,+1]$ there is an* efficient *blockwise p-tampering attack that increases the average of f over a sampled input by at least $\Omega(p) \cdot \mathrm{Var}[f(\overline{X})]$.*

See Theorem 4 for a formalization. Similarly to [1], we also prove a variant of Theorem 1 for the special case of Boolean functions, but with better parameters (see Theorem 5). However, some of the applications of this biasing lemma (e.g., for attacking cryptographic primitives, or attacking learning algorithms with non-Boolean cost/loss functions) we need to use the non-Boolean attack of Theorem 1.

Our main biasing p-tampering attack on bounded functions even applies to the settings where $\overline{X}$ is *not* a product distribution. In that case, we assume

[2] We only allow the tampering algorithm to produce something in the support set. A more general definition allows the tampering algorithm to make choices out of the support set, however, our restriction only makes our attacks stronger.

that $\overline{X}$ is sampled in a 'stateful' way, and that the next block X_i is sampled conditioned on adversary's choices of blocks. This extension allows our model to include previous special models of p-tampering attacks against random walks on graphs [3].

We also prove some applications for our main biasing attack that rely on the *blockwise* nature of it. In addition to obtaining attacks against the security of cryptographic primitives as well as multi-source randomness extractors through blockwise p-tampering, we also demonstrate applications beyond cryptography. In particular, by relying on the power of biasing attacks over *non-uniform* distributions, we show how to attack and increase the error of *learning* algorithms that output classifiers, through an attack that injects a p fraction of adversarial data in an online way. In what follows we briefly discuss each of these applications.

Attacks on Randomness of Cryptographic Primitives. As mentioned, the bitwise p-tampering attack of [1] for biasing functions was at the core of their attacks breaking the security of cryptographic primitives by tampering with their randomness. By using our biasing attack of Theorem 1 we immediately obtain blockwise attacks against the same primitives. This time, our attacks work *regardless* of how randomness is packed into blocks, and is also 'robust' in the sense that the attack succeeds even if the tampering probabilities $p_1, p_2, \ldots$ are *not* equal so long as $p \leq p_i$ for all i.[3]

Corollary 1 (Informal). *Let $\mathcal{P}$ be one of the following primitives. CPA secure public-key or private-key encryption, efficient-prover zero-knowledge proofs for* **NP***, commitment schemes, or two party computation where only one party gets the output. Then there is an efficient blockwise p-tampering attack that breaks the security of $\mathcal{P}$ with advantage $\Omega(p)$. In particular, the attack succeeds even if the length of the tampered randomness blocks are* unknown *a priori and only become clear during the attack.*

The above theorem could be obtained by plugging in our biasing attack of Theorem 1 into the proofs of [1].

Achieving security against blockwise p-tampering? In addition to presenting the power of bitwise p-tampering attacks, the work of [1] also showed how to achieve secure protocols against bitwise p-tampering attacks for 'forging-based' primitives such as signatures for $p = 1/\text{poly}(\kappa)$ where κ is the security parameter. For the same primitives, when we move to the blockwise setting, whether or not achieving positive (secure) results is possible *depends* on the block sizes of the tampering attack. For example, if the *whole* randomness of the key generation algorithm of a signature scheme becomes tamperable as a single

[3] In fact, we observe that the bitwise p-tampering attack of [1] is also robust, but proving robustness becomes more challenging for our blockwise p-tampering attack. Moreover, we believe robustness is an important feature for cryptographic attacks and so worth to be studied explicitly, as some attacks, e.g., the reduction from bitwise to blockwise p-tampering (Please see the full version for the proof.), are not necessarily robust.

block (with probability $p \geq 1/\text{poly}(\kappa)$) the adversary can choose an insecure key. On the other hand, if all the blocks are of constant size (or even of size $o(\lg \kappa)$) similar arguments to those in [1] could be used to make 'forging-based' primitives secure for any $p \leq \kappa^{-\Omega(1)}$.

Efficient Attacks for Biasing Extractors. Our blockwise p-tampering attacks for biasing functions are natural tools for 'biasing attacks' against (seedless) randomness extractors from block sources.

Biasing Multi-source Seedless Extractors. We can directly use our p-tampering attacks against *any* specific, multi-source, seedless randomness extractors [12,39,43]. Namely, suppose f is an efficient seedless extractor who takes n blocks of randomness $(x_1, \ldots, x_n) \leftarrow (X_1 \times \cdots \times X_n)$ where the distribution X_i belongs to a class of randomness source. Then, for any choice of samplable $\overline{X} = (X_1, \ldots, X_n)$, Theorem 5 gives an efficient p-tampering attacker who could transform the distribution $\overline{X}$ into $\overline{Y}$ such that $|\mathbb{E}[f(\overline{Y})]| \geq \Omega(p)$. Note that the interesting aspect of $\overline{Y}$ is that it is identical to $\overline{X}$ in $(\approx 1 - p)$ fraction of the blocks. In particular, as we will see, our attacker of Theorem 1 has the property that upon tampering with each block, all it does is to either leave as is or 'resample' it once.

The second application of our p-tampering attacks against extractors is different in the sense that instead of attacking extractors when unbiased extraction is possible, it gives an alternative algorithmic proof for a known impossibility result [6,19,22,36] regarding block Santha-Vazirani sources [39]. Below, by $U_i^j = U_i \times \cdots \times U_i$ we refer to j blocks each consisting of i uniform bits.

Impossibility of Randomness Extraction from SV Sources. The celebrated work of Santha and Vazirani [39] proved a strong negative result about deterministic randomness extraction from sources with high min-entropy. An SV source (see Definition 7) is a joint distribution $(X_1, \ldots, X_n)$ over $\{0,1\}^n$ with the guarantee that every bit is δ-close to uniform even if we condition on all the previous bits. In particular, [39] proved that for any deterministic (supposedly extractor) function $f\colon \{0,1\}^n \mapsto \{+1,-1\}$, there is always an δ-SV source $\overline{X} = (X_1, \ldots, X_n)$ such that $|\mathbb{E}[f(\overline{X})]| \geq \Omega(\delta)$. The work of Reingold et al. [36] gave an elegant simple proof for this result using the so called 'half-space' sources, and this idea found its way into the work of Dodis et al. [19] where they generalized the result of [39] to *block* sources [13]. A (ℓ, k)-block SV source is a sequence of blocks of length ℓ bits such that each block has min-entropy at least k conditioned on previous blocks (see Definition 8).

Even though p-tampering attacks do *not* generate block-SV sources with 'high' min-entropy in general, we show that the *specific* p-tampering attacker of our Theorem 1 does indeed generate an $(\ell, \ell - p)$ block-SV source. As a result, we get an alternative proof for the impossibility of deterministic extraction from block-SV sources, but this time through *efficient* p-tampering attacks.[4] In particular, we prove the following.

[4] Note that this is indeed a stronger condition than just getting a samplable source. See Remark 1.

Theorem 2 (Efficient p-tampering attacks over block SV sources). *Let the function $f\colon \{0,1\}^{\ell \cdot n} \mapsto \{+1,-1\}$ be a 'candidate' efficient deterministic extractor for $(\ell, \ell - p)$ block SV sources. Then there is an efficient p-tampering attack that generates a $(\ell, \ell - p)$ block SV source for which the average of f becomes $\Omega(p)$.*

Our main contribution in Theorem 2 is the efficiency of its p-tampering attacker, as without that condition one can prove Theorem 2 using a *computationally unbounded* p-tampering attacker and the proof implicit in [19,36] and explicit in [6,22] for the case of block SV sources. In fact, we prove a more general result than Theorem 2 by proving the impossibility of efficient bit bit extractors from yet another generalization of SV sources, called mutual max-divergence [23] (MMD) sources (see Definition 6).

Attacking Learners. In this work, we also use our blockwise p-tampering attack in the context of "adversarial" machine learning [5,35] where an adversary aims to increase the error of a learning algorithms for a specific test data that is known to him. In what follows, the reader might find the review of the standard terminology at the beginning of Sect. 4.2 useful.

Targeted poisoning attacks against learners. Poisoning attacks (a.k.a causative attacks) [2,40,44] model attacks against learning systems in which the adversary manipulates the training data $\overline{x} = (x_1, \ldots, x_n)$, where x_i is the i'th *labeled* training example, in order to increase the error of the learning algorithm. Poisoning attacks could model scenarios where the tampering happens *over time* [37,38] e.g., because the learning algorithm is "retrained" daily or weekly using potentially tamperable data. *Targeted* (poisoning) attacks [40] refer to the setting where the adversary *knows* a specific test data $\mathcal{X}$ over which the hypothesis will be tested, and she probably has some interest in increasing the error of the hypothesis over that particular test set $\mathcal{X}$. For simplicity of discussion, below we assume that $\mathcal{X} = \{(d,t)\}$ where t is the label of d and the adversary's goal is to make the learning algorithm output a wrong label for d.

A very natural model for how the poisoning attacks occur was defined by Valiant [42]. In this model, a training oracle $O_X(.)$ for a distribution X (from which the training sequence $\overline{x} = (x_1, \ldots, x_n)$ will be sampled) would be manipulated by an adversary as follows. Whenever the training algorithm queries this oracle, with probability $1 - p$ the answer is generated from the original oracle O_X and with probability p a stateful adversary A gets control over the oracle and answers with an arbitrary pair (d,t). Many subsequent work (e.g., [10,31]) studied how to make learners secure against such noise but *not* in the targeted setting.

Valiant's model vs p-tampering. Valiant's adversarial model for the training oracle is indeed very similar to our blockwise p-tampering model except for the fact that in the Valiant's model, the adversary is allowed to use wrong labels (i.e., $x_i = (d,t)$ where t is *not* the correct label for d). However, as we discussed above, our p-tampering attackers are not allowed to go out of the 'support set' of the distribution (see Definition 18). In this work, we prove the

following attack against deterministic learners of classifiers (see Theorem 8 for a formalization). One subtle difference between the models is that in Valiant's model, the adversary knows everything about the *current* state of the learner, while in our model, the adversary knows the history of the blocks. For all of our attacks, all adversary needs is to 'continue' the computation done by the learner, and knowing the current state (as in Valiant's model) allows us to do so, even if the previous blocks are unknown. Therefore, all of our p-tampering attacks indeed apply in Valiant's model.

Theorem 3 (Informal–Targeted poisoning attacks against classifiers). *Let L be a* deterministic *learning algorithm L that takes a sequence $\overline{x} = (x_1, \ldots, x_n)$ of i.i.d samples from the same distribution X, where $x_i = (d_i, \ell_i)$ and ℓ_i is the label of d_i. Suppose, without tampering, the probability of L making a mistake on test example d is δ over the choice of $x_1, \ldots, x_n \leftarrow X$. Then there exists a p-tampering attack over the training sequence $(x_1, \ldots, x_n)$ that increases the error for classifying d to $\delta' \geq \delta + \Omega_\delta(p)$. Moreover, if X is efficiently samplable, the attack is efficient as well.*

Note that the above attacker is a p-tampering one, meaning it never goes out of the support set of the distribution. In other words, our attacker does *not* use any wrong labels in its adversarial samples! Therefore, our attacks are 'defensible' in the sense that what they produce is always a possible legitimate outcome of the honest sampling, so it could not be *proved* in court that the data is not generated honestly! Previous work on poisoning attacks (e.g., [2,40,44]) has studied attacks against *specific* learners, while our result can be applied to *any* learner.

Comparison with the distribution-independent setting of [10,31]. Previous works of Kearns and Li [31] and Bshouty et al. [10] have already proved impossibility of PAC learning in Valiant's model of adversarial noise. In addition to using wrong label in their attacks (which is not allowed in the p-tampering model) there is also another distinction between their model and our p-tampering poisoning attacks. The attacks of [10,31] are proved in the *distribution-independent* setting, and their negative results heavily rely on the *existence* of some initial distribution that is not PAC learnable under adversarial noise. Our attacks, on the other hand, apply even to the *distribution-specific* setting, where the adversary has no control over the initial distribution, and it can always turn that distribution against the learner.

1.2 Ideas Behind Our Blockwise p-Tampering Biasing Attack

In this subsection we describe some of the ideas behind the proof of our Theorem 1.

Reduction to bitwise tampering? Our first observation is that blockwise $\widetilde{p}$-tampering over *uniformly* distribute blocks $U_{s_1} \times \ldots U_{s_n}$ could be reduced to p-tampering over $N = \sum_i s_i$ many uniform bits, as long as $1 - \widetilde{p} \leq (1-p)^{s_i}$

for every s_i. The idea is that if $1 - \widetilde{p} \leq (1-p)^{s_i}$, then the probability of the whole block U_{s_i} getting tampered with in the blockwise model is at least the probability that *at least* one of the bits are tampered with in the bitwise model. Therefore, a blockwise attacker can 'emulate' the bitwise attacker internally. See the full version for a formalization of this argument.)

However, this reduction is imperfect in three aspects. (1) Firstly, to use this reduction we will need to use $p \approx \widetilde{p}/s$ where s is the maximum length of any block. Therefore, we cannot gain any bias more than $1/s$ which, in particular, would be at most $o(1)$ for non-constant block sizes $s = \omega(1)$. This prevents us from getting applications (e.g., attacks against extractors) that require large $\Omega(1)$ bias. (2) Secondly, this reduction only works for blocks that are originally distributed as uniform bits (i.e., U_s), and so it cannot be applied to general non-uniform distributions, which is indeed the setting of our p-tampering attacks against learners. (3) Finally, this reduction does not preserve robustness as the $\widetilde{p}$-tampering algorithm would need to know the *exact* probabilities under which the tampering happens, while in our applications of blockwise tampering to cryptographic primitives robustness we aim for robust attacks that do not depend on this exact knowledge. Because of all this, in this work we aim for a direct attack analyzed in the blockwise regime.

The work of [1] used a so called 'mild-greedy' attack for biasing real-valued bounded function in a bitwise p-tampering attack. Roughly speaking, this attack works as follows. When the tampering happens, the tampering algorithm first picks a random bits b'_i. Then, using a random continuation $b'_{i+1}, \ldots, b'_n$ it interprets $s = f(b_1, \ldots, b_{i-1}, b'_i, \ldots)$ as how good the choice of b'_i is. Then, using a biased coin based on s, the tampering algorithm either keeps b'_i or it flips it to $1 - b'_i$. This attack, unfortunately, is tailored of the bitwise setting, as flipping a block is not natural (or even well defined).

Our new one rejection sampling attack. In this work propose a new attack for the blockwise setting that is inspired by the mild-greedy attack of [1]. Our attack is not exactly a 'generalization' of the mild-greedy attack to the blockwise setting, as even for the case of uniform blocks of one bit, it still differs from the mild-greedy attack, but it is nonetheless inspired by the one-greedy attack and its analysis also uses ideas from the analysis of mild-greedy attack [1]. We call our tampering attack *one rejection sampling*, denoted by ORSam, and it works as follows: given previously chosen blocks $(y_1, \ldots, y_{i-1})$ for $\overline{X}$ (some of which might be the tampered blocks) the tampering algorithm ORSam first samples $(y'_i \leftarrow X_i, \ldots, y'_n \leftarrow X_n)$ 'in its head', then gets $s = f(y_1, \ldots, y_{i-1}, y'_i, \ldots, y'_n)$, and outputs:

$$\begin{cases} \text{Case 1: with probability } \frac{1+s}{2} : \text{keep } y'_i \\ \text{Case 2: with probability } \frac{1-s}{2} : \text{use a fresh sample } y''_i \leftarrow X_i. \end{cases}$$

Why does one-rejection sampling work? The main challenge is to show that the simple one-rejection sampling attack described above actually achieves bias proportional to the variance. In order to relate the bias to the variance of the

function, we first need to define two notations. For every prefix $x_{\leq i} = x_1, \ldots, x_i$ let $\hat{f}[x_{\leq i}] = \mathbb{E}[f(\overline{X})|X_1 = x_1, \ldots, X_i = x_i]$ to be the average of function f w.r.t to distribution $\overline{X}$ conditioned on that prefix. Also let $g[x_{\leq i}] = \hat{f}[x_{\leq i}] - \hat{f}[x_{\leq i-1}]$ be the change in average of f (i.e., $\hat{f}$) when we go from $x_{\leq i-1}$ to $x_{\leq i}$. A straightforward calculation shows that

$$\mathrm{Var}[f(\overline{X})] = \mathop{\mathbb{E}}_{(x_1,\ldots,x_n)\leftarrow\overline{X}} \Big[\sum_{i\in[n]} g[x_{\leq i}]^2 \Big] = \sum_{i\in[n]} \mathop{\mathbb{E}}_{x_{\leq i}\leftarrow(X_1,\ldots,X_i)} \big[g[x_{\leq i}]^2\big]. \quad (1)$$

That is simply because the sequence $(\hat{f}[x_{\leq 0}], \ldots, \hat{f}[x_{\leq n}])$ forms a martingale. Suppose $\overline{Y} = (Y_1, \ldots, Y_n)$ is the new distribution after the p-tampering happens over $\overline{X}$. Equation (1) suggests the following natural idea for lower bounding the amount of "global gain" that is achieved for increasing the average $d = \mathbb{E}[f(\overline{Y})] - \mathbb{E}[f(\overline{X})]$ under the attack's generated distribution by relating it to the variance $\mathrm{Var}[f(\overline{X})]$. In particular, it would suffice to lower bound the "local gains" for average of f when we apply our *one rejection sampling* with probability p for a particular block i, by relating it the term $\mathbb{E}_{(x_1,\ldots,x_n)\leftarrow\overline{X}}[g[x_{\leq i}]^2]$ (for the same fixed i). Direct calculation shows that the 'local gain' obtained by our one-rejection sampling attack for any prefix $x_{\leq i}$ is *exactly* $\frac{p}{2} \cdot \mathbb{E}_{x_{i+1}\leftarrow X_{i+1}}[g[x_{\leq i}, x_{i+1}]^2]$.

Unfortunately, a subtle point prevents us from using the above argument, because as soon tampering happens, we *deviate* from the original distribution $\overline{X}$, and the 'prefixes' of the blocks come from a new distribution $\overline{Y}$ rather than $\overline{X}$, so we cannot directly use to Eq. (1) to lower bound the local gains by relating them to $\mathrm{Var}[f(\overline{X})]$. Nonetheless, it can be shown that a variant of Eq. (1) still holds in which, roughly speaking, $\mathrm{Var}[f(\overline{Y})]$ substitutes $\mathrm{Var}[f(\overline{X})]$. Therefore, it would be sufficient to lower bound $\mathrm{Var}[f(\overline{Y})]$ based on $\mathrm{Var}[f(\overline{X})]$. For this goal, we employ similar ideas to those of [1] to show by induction over i that at any moment during the attack *either* the average *or* the variance of $\hat{f}[x_{\leq i}]$ under the *new* tampered distribution $\overline{Y}$ is large enough. See Sect. 5 for more details.

1.3 Further Related Work and Models

Since the work of Boneh et al. [9] it is known that even *random* tampering with computation of certain protocols could lead to devastating attacks. The work of Gennaro et al. [26] initiated a formal study of algorithmic tamper resilience. Along this direction, non-malleable codes, introduced by Dziembowski et al. [25], become a central tool for preventing tampering attacks on the internal state of an algorithm. More recently, Chandran et al. [11] studied non-malleable codes in the *blockwise* tampering model that bears similariteis to our model in this work, though our goals are completely different. Finally, Bellare et al. [7] initiated the study of *algorithm substitution* attacks where a powerful attacker can adversarially substitute components of the algorithm.

Coin-tossing. At a high level, our blockwise tampering attacks, specially for biasing Boolean functions, have some conceptual similarities to attacks against

coin-tossing protocols [8,15,17,29,34]. Indeed, both types of attacks aim at biasing a final bit by 'substituting' some 'blocks'. In our setting, the block is the next sampled chunk of randomness, and for coin tossing blocks are maliciously chosen messages to the other party! However, the pattern of tampering in such attacks is one out of two complementing sets (referring to each party's turns), while in our setting each block becomes tamperable with an independent probability p.

Tampering with 'bounded budget'. The works of [15,27,33] studied the power of related tampering attacks in the blockwise setting where the goal of the adversary is indeed to bias the output of a function. However, in these papers, while the adversary has a 'limited budged' of how many times to tamper, it *can choose* when to tamper with a block, while, in our model the adversary will have no control on about $1-p$ fraction of the blocks, and he does not get to choose which blocks will be so. The work of Dodis [20] studies a 'mixture' of both models where the adversary has a bounded budged that she can use upon choice, but she also gets to tamper 'randomly' otherwise.

2 Preliminaries

Logarithms are denoted by $\lg(\cdot)$ and, unless specified otherwise, they are in base 2. By $a, b \in \mathcal{D}$ we mean that $a \in \mathcal{D}$ and $b \in \mathcal{D}$. For a string $x \in \{0,1\}^*$, by $|x| = n$ we denote that $x \in \{0,1\}^n$. For a randomized algorithm S, we only explicitly represent its input and do not represent its randomness and by $y \leftarrow S(x)$ we denote the process of running $S(x)$ using fresh randomness and getting y as output.

Notation on distributions and random variables. Unless specified otherwise, all of the random variables and distributions in this work are discrete and finite. We use uppercase letters to denote random variables and distributions (e.g., X). For real valued random variable X, by $\mathbb{E}[X]$ and $\mathrm{Var}[X]$, we mean (in order) the expected value and variance of X. We usually use the same letter to refer to distributions and random variables sampled from them. By $\mathrm{Supp}(X) = \{x \mid \Pr[X = x] > 0\}$ we denote the support set of X. The process of sampling x from X is denoted by $x \leftarrow X$ and $X \equiv Y$ is used to show that X and Y are distributed identically.

By U_m we denote the random variable uniformly distributed over $\{0,1\}^m$. By (X, Y) we denote random variables X, Y that are distributed jointly. By $(X \times Y)$ we mean (X, Y) where X and Y are independently sampled from their marginal distribution. For joint random variables (X, Y) and for any $y \leftarrow Y$, by $(X \mid y)$ we denote the distribution of X conditioned on $Y = y$. By using a random variable like X in an expected value (or probability) we mean that the expected value (or the probability) is also over X (e.g., $\mathbb{E}[f(X)] = \mathbb{E}_{x \leftarrow X}[f(x)]$ and $\Pr[f(X) = 1] = \Pr_{x \leftarrow X}[f(x) = 1]$). We also use the tradition that the multiple appearances of the same random variable X in the same phrase refer to identical samples (e.g., it always holds that $\Pr[X = X] = 1$). For a random variable D, we also use $D(x)$ to denote $\Pr[D = x]$.

Definition 1 (Bit extraction). *Let $\mathcal{X}$ be a set of distributions over a domain $\mathcal{D}$. We call a function $f\colon \mathcal{D} \mapsto \{+1,-1\}$ an* ε-extractor *for $\mathcal{X}$ (sources) if for every $X \in \mathcal{X}$ it holds that $|\mathbb{E}[f(X)]| \leq \varepsilon$.*

Definition 2. $\mathrm{H}_\infty(X) = \min_{x \in \mathrm{Supp}(X)} \lg(1/p(x))$ *is the* min-entropy *of X.*

Definition 3 (Span of distributions). *Let $\mathcal{X} = \{X_1, \ldots, X_k\}$ be a set of distributions over the same domain. For $\alpha_1 + \cdots + \alpha_k = 1$, by $X = \sum_{i\in[k]} \alpha_i X_i$ we refer to the distribution X such that $\Pr[X = a] = X(a) = \sum_i \alpha_i X_i(a)$. Namely, X can be sampled by the following process: first sample $i \in [k]$ with probability α_i, then sample $x \leftarrow X_i$ and output x. The* span *of distributions in $\mathcal{X}$ is defined to be the set of all convex combinations of distributions in $\mathcal{X}$:* $\mathrm{Span}(\mathcal{X}) = \{X = \sum_{i\in[k]} \alpha_i X_i \mid \sum_{i\in[k]} \alpha_i = 1\}$.

Lemma 1 (Hoeffding's inequality). *Suppose $A_1, \ldots, A_n$ are i.i.d random variables distributed over $[-1,+1]$ with expected value $\mathbb{E}[A_i] = \mu$, and let $A = \mathbb{E}_{i\leftarrow[n]}[A_i]$ be their average. Then, for all $\varepsilon \geq 0$ we have $\Pr[|A-\mu| \geq \varepsilon] \leq e^{-n\cdot\varepsilon^2/2}$.*

2.1 Distance Measures

Definition 4 (Statistical distance). *The* statistical distance *(a.k.a. total variation distance) between random variables X, Y is defined as*

$$D_{\mathsf{SD}}(X,Y) = \max_{E \subseteq \mathrm{Supp}(X)} \Pr[X \in E] - \Pr[Y \in E].$$

The following lemma gives a well known characterization of the statistical distance.

Lemma 2 (Characterizing statistical distance). *It holds that $D_{\mathsf{SD}}(X,Y) \leq p$ iff there are distributions Z, X', Y' such that $X = (1-p)Z + pX'$ and $Y = (1-p)Z + pY'$. In particular, if $Y = (1-p)X + pZ$ then we have $D_{\mathsf{SD}}(X,Y) \leq p$ because it always holds that $X = (1-p)X + pX$.*

Definition 5 (KL-divergence). *The* Kullback-Leibler (KL) divergence *from distribution Q to distribution P is defined as follows: $D_{\mathsf{KL}}(P||Q) = \mathbb{E}_{a\leftarrow P} \lg(P(a)/Q(a))$ if $\mathrm{Supp}(P) \subseteq \mathrm{Supp}(Q)$, and $D_{\mathsf{KL}}(P||Q) = \infty$ if $\mathrm{Supp}(P) \not\subseteq \mathrm{Supp}(Q)$.*

Definition 6 (Max-divergence[23]). *The* max-divergence *from Q to P is defined as follows: $D_\infty(P||Q) = \max_{a\in\mathrm{Supp}(P)} \lg(P(a)/Q(a))$ if $\mathrm{Supp}(P) \subseteq \mathrm{Supp}(Q)$, and if $\mathrm{Supp}(P) \not\subseteq \mathrm{Supp}(Q)$, then $D_\infty(P||Q) = \infty$.*

The work of [23] defined the notion of max-divergence using $\mathbf{e}$ as the base for logarithm, but in this work we use a variation of it using base 2, which is the same up to a multiplicative constant factor $\lg \mathbf{e}$. The following lemma lists some of the basic properties of max-divergence (see Definition 6).

Lemma 3 (Properties of max-divergence). *Let X, Y be distributions and $p < 1$.*

1. *The following conditions are equivalent.*
   (a) $D_\infty(X||Y) \leq \lg(1/(1-p))$.
   (b) *For all* $a \in \mathrm{Supp}(X)$ *it holds that* $\Pr[X = a] \cdot (1-p) \leq \Pr[Y = a]$.
   (c) *There exists some random variable* Z *such that* $Y = (1-p)X + pZ$. *Namely, Y can be sampled as: with probability $1-p$ sample from X and with probability p sample from Z.*
2. *For* $\mathrm{Supp}(Y) \subseteq \{0,1\}^m$, $\mathrm{H}_\infty(Y) \geq k$ *iff* $D_\infty(Y||U_m) \leq m - k$.
3. *If* $D_\infty(X||Y) \leq r$ *and* $D_\infty(Y||X) \leq r$, *then* $D_{\mathsf{KL}}(X||Y) \leq r(2^r - 1)$.

Proof (Proof Sketch). Here we only sketch the proofs as they are straightforward. The equivalence of Parts 1a and 1b directly follows from the definition of max-divergence, so here we only show the equivalence of Parts 1b and 1c. Assuming Part 1c we have

$$\Pr[X = a] \cdot (1-p) \leq \Pr[X = a] \cdot (1-p) + \Pr[Z = a] \cdot p = \Pr[Y = a]$$

which implies Part 1b. Assuming Part 1b, we define the distribution Z over $\mathrm{Supp}(Y)$ as follows: $Z(a) = (Y(a) - (1-p) \cdot X(a))/p$. It is easy to see that $Z(a) \geq 0$ and that $\sum_a Z(a) = 1$, so Z indeed defines a distribution. Moreover, we have

$$X(a) \cdot (1-p) + Z(a) \cdot p = X(a) \cdot (1-p) + (Y(a) - X(a) \cdot (1-p)) = \Pr[Y = a]$$

which implies that $Y = (1-p)X + pZ$, proving Part 1c.

Part 2 directly follows from the definitions of min-entropy and max-divergence.

Part 3 follows from the same proof give in [23] but using the logarithm base 2 istead of $\mathbf{e}$ in the definition of max-divergence.

2.2 Santha-Vazirani Sources and Their Generalizations

Definition 7 (SV sources [39]). *A joint distribution $\overline{X} = (X_1, \ldots, X_n)$ where $X_i \in \{0,1\}$ for all $i \in [n]$ is a δ-Santha-Vazirani (δ-SV) source with bias at most $\delta \in [0,1]$, if for all $i \in [n]$ and all $x_1, \ldots, x_i \in \{0,1\}$ it holds that $(1-\delta)/2 \leq \Pr[X_i = x_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \leq (1+\delta)/2$.*

The following definition is a close variant of Block SV Sources defined in [13] where we allow the blocks to have different lengths and specify the amount of *loss* in the min-entropy (compared to the uniform distributing) in each block.

Definition 8 (Block SV Sources [13]). *Suppose $\overline{X} = (X_1, \ldots, X_n)$ is a joint distribution where $X_i \in \{0,1\}^\ell$ for all $i \in [n]$. We call $\overline{X}$ a (ℓ, k)-block SV source if for all $i \in [n]$ and all possible $(x_1, \ldots, x_{i-1}) \leftarrow (X_1, \ldots, X_{i-1})$ it hold that $\mathrm{H}_\infty(X_i \mid x_1, \ldots, x_{i-1}) \geq k$.*

It is easy to see that a δ-SV source is a $(1, 1-\gamma)$-block-SV source for $\gamma = \lg(1+\delta) \leq \delta$. The following definition by Beigi et al. [6] generalizes both of the above definitions of SV and Block-SV sources.

Definition 9 (Generalized SV Sources [6]). *Let $\mathcal{D}$ be a set of distributions (dices) over alphabet C. A distribution $\overline{X} = (X_1, \ldots, X_n)$ over C^n is a* Generalized SV source *w.r.t $\mathcal{D}$ if for all $i \in [n]$ and $x_1, \ldots, x_{i-1} \in C$ there exists $S \in \mathrm{Span}(\mathcal{D})$ such that for all $x_i \in C$ it holds that*

$$\Pr[X_i = x_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] = \Pr[S = x_i].$$

3 Blockwise p-Tampering: Definitions and Main Results

In this section, we will describe our results formally.

Notation on sequences of random variables. By D^n we denote the product distribution $D \times \cdots \times D$ (n times). Using this notation, by U_m^n we mean a sequence of n *blocks* each distributed independently like U_m. Thus, although both of U_m^n and U_n^m are eventually $m \cdot n$ random bits, one is divided into n blocks and one is divided into m blocks. For a vector $x = (x_1, \ldots, x_n)$ we let $x_{\leq i} = (x_1, \ldots, x_i)$, $x_{<i} = (x_1, \ldots, x_{i-1})$.

Definition 10 (Valid prefixes and conditional sampling). *Let $\overline{X} = (X_1, \ldots, X_n)$ be a joint distribution. We call $x_{\leq i} = (x_1, \ldots, x_i)$ a* valid prefix *for $\overline{X}$ if there are $x_{i+1}, \ldots, x_n$ such that $(x_1, \ldots, x_n) \in \mathrm{Supp}(\overline{X})$ (i.e., $x_{\leq i} \in \mathrm{Supp}(X_{\leq i})$). We use $\mathrm{ValPref}(\overline{X})$ to denote the set of all valid prefixes of $\overline{X}$ (including the empty string $x_{\leq 0}$). For a valid prefix $y_{\leq i} \in \mathrm{ValPref}(\overline{X})$, by $(X_i \mid y_{\leq i-1})$ we denote the conditional distribution $(X_i \mid X_1 = y_1, \ldots, X_{i-1} = y_{i-1})$.*

Definition 11 (Online-samplable sequences of random variables). *We call a* randomized *algorithm $S(\cdot)$ an* online sampler *for a joint distribution. Let $\overline{X} = (X_1, \ldots, X_n)$ if for every valid prefix $x_{\leq i-1} \in \mathrm{ValPref}(\overline{X})$, it holds that $S(x_{\leq i-1})$ outputs according to $(X_i \mid x_{\leq i-1})$. If $\overline{X} = \overline{X}^{(n)}$ is a vector from a* family *of vectors indexed by n, we let $N = N(n)$ be the total length of the representation of $\overline{X}$ (i.e., $(X_1, \ldots, X_n) \in \{0,1\}^N$) and assume that n could be derived from $N(n)$. In that case, an online sampler $S(\cdot)$ for $\overline{X}^{(n)}$ takes also N as input and it holds that $S(1^N, x_{\leq i-1}) \equiv (X_i \mid x_{\leq i-1})$. We call $\overline{X} = \overline{X}^{(n)}$* efficiently online samplable *if there exists an online sampler S for $\overline{X}$ that runs in polynomial time (i.e. $\mathrm{poly}(N)$). When n is clear from the context we might simply drop 1^N and simply write $S(x_{\leq i-1})$.*

Definition 12 (Tampering algorithms for sequences of random variables). *Let $\overline{X} = (X_1, \ldots, X_n)$ be an arbitrary joint distribution. We call a (potentially randomized and even computationally unbounded) algorithm Tam an (online)* tampering algorithm for $\overline{X}$ *if given any valid prefix $x_{\leq i-1} \in \mathrm{ValPref}(\overline{X})$, $\mathsf{Tam}(x_{\leq i-1})$ always outputs x_i such that $x_{\leq i} \in \mathrm{ValPref}(\overline{X})$.*

If $\overline{X} = \overline{X}^{(n)}$ *is a vector from a* family *of vectors indexed by* n*, we call* Tam *an* efficient *tampering algorithm for* $\overline{X}$ *if it runs in time* $\mathrm{poly}(N)$ *where* $N = N(n)$ *is the total bit length of the vector* $\overline{X}$ *(i.e.,* $(X_1, \ldots, X_n) \in \{0,1\}^N$*).*

Note that in Definition 12, we only allow the tampering algorithm to produce something in the support set of the joint distribution.

The following definition defines a notation for representing the "chances" that might be given to a tampering algorithm to tamper with the joint distribution $\overline{X} = (X_1, \ldots, X_n)$. We need this generalization to formally define the robustness of p-tampering attack when p changes during the attack.

Definition 13 (Probability trees over sequences of random variables). *Let* $\overline{X} = (X_1, \ldots, X_n)$ *be an arbitrary joint distribution. We call a function* $\rho\colon \mathrm{ValPref}(\overline{X}) \mapsto [0,1]$ *a* probability tree *over* $\overline{X}$*. For* $0 \le p \le q \le 1$*, we call* $\rho[\cdot]$ *a* $[p,q]$*-probability tree over* $\overline{X}$ *if* $\rho(x_{\le i}) \in [p,q]$ *for all* $x_{\le i} \in \mathrm{ValPref}(\overline{X})$*. We call* $\rho[\cdot]$ *the* p*-probability tree over* $\overline{X}$ *if* $\rho[x_{\le i}] = p$ *for all* $x_{\le i} \in \mathrm{ValPref}(\overline{X})$*.*

Now we define the outcome of an actual "tampering game" in which a tampering algorithm gets to tamper with a joint distribution $\overline{X} = (X_1, \ldots, X_n)$ according to some probability tree defined over $\overline{X}$.

Definition 14 (ρ-tampering variations of distributions). *Let* $\overline{X} = (X_1, \ldots, X_n)$ *be an arbitrary joint distribution, and let* $\rho[\cdot]$ *be a probability tree over* $\overline{X}$*. We say that a tampering algorithm* Tam *for* $\overline{X}$ *generates* $\overline{Y}$ *from* $\overline{X}$ *through a* ρ*-tampering attack if* $\overline{Y} = (Y_1, \ldots, Y_n)$ *is inductively sampled as follows. Given any valid prefix* $y_{\le i-1} \in \mathrm{ValPref}(\overline{Y})$ *we will sample* Y_i *through the following process:*

- *with probability* $1 - \rho[y_{\le i-1}]$*, sample* Y_i *from* $(X_i \mid X_{\le i-1} = y_{\le i-1})$*, and*
- *with probability* $\rho[y_{\le i-1}]$*, sample* $Y_i \leftarrow \mathsf{Tam}(y_{\le i-1})$*.*

Equivalently, using Definition 3, for all $y_{\le i-1} \in \mathrm{ValPref}(\overline{Y})$ *we have* $(Y_i \mid y_{\le i-1}) = (1 - \rho[y_{\le i-1}]) \cdot (X_i \mid X_{\le i-1} = y_{\le i-1}) + \rho[y_{\le i-1}] \cdot \mathsf{Tam}(y_{\le i-1})$*. In this case, we also call* $\overline{Y}$ *a* ρ-tampering variation *of* $\overline{X}$*. In case* ρ *is the constant function* p*, we call* $\overline{Y}$ *a* p*-tampering variation of* $\overline{X}$ *and we say that* Tam *generates* $\overline{Y}$ *from* $\overline{X}$ *through a* p*-tampering attack.*

Note that even in cases where we end up sampling Y_i from the "untampered" distribution of X_i (which happens with probability at least $1 - \rho[x_{\le i-1}]$) we still sample from X_i conditioned on the *possibly tampered* prefix $(y_1, \ldots, y_i)$. In other words, the result of the tampering algorithm determines, in case it happens, will completely substitute the tampered block and the sampling will continue as if the history of the blocks were from the untampered sequence $X_1, \ldots, X_i$. For the special case that X_i's are independent distributions (e.g., when $\overline{X}$ is uniform distribution over some set Σ^n) we will not need to do this.

Prefixes remain valid. Note that because in Definition 14 the algorithm Tam is a (valid) tampering algorithm for $\overline{X}$, all the resulting prefixes will remain valid

for $\overline{X}$ and we will have $\mathrm{ValPref}(\overline{Y}) \subseteq \mathrm{ValPref}(\overline{X})$. In fact, we get $\mathrm{ValPref}(\overline{Y}) = \mathrm{ValPref}(\overline{X})$ if $\rho[x_{\leq i}] < 1$ for all $x_{\leq i} \in \mathrm{ValPref}(\overline{X})$. A more general definition of tampering algorithms (compared to Definition 12) could use a larger support set $\mathcal{Z}$ where $\mathrm{ValPref}(\overline{X}) \subset \mathcal{Z}$ and only require the tampering algorithm to produce prefixes in $\mathcal{Z}$. However, since our main contributions in this paper is to give attacks, by restricting our model to require the attackers to remain in $\mathrm{ValPref}(\overline{X})$ only makes our results stronger.

Remark 1 (Efficient tampering vs. efficient sampling). Note that an *efficient tampering* refers only to when the algorithm Tam is polynomial time, and it can apply even to settings where $\overline{X}$ and its variation generated by Tam are *not* efficiently samplable. On the other hand, using the standard terminology, $\overline{X}$ is efficiently samplable if one can efficiently sample *all* of the blocks of $\overline{X}$ *simultaneously*. Of course, if $\overline{X}$ is efficiently *online* samplable and if Tam is also an efficient tampering for $\overline{X}$, then the variation $\overline{Y}$ of $\overline{X}$ produced by tampering attack Tam will also be trivially efficiently online-samplable, but we emphasize that this is a specific way of getting an efficient sampler for $\overline{Y}$, and so the efficiency of our tampering attacks shall not be confused with mere efficient samplablility of the final distribution $\overline{Y}$.

Remark 2 (An alternative definition). An alternative variant of Definition 14 could 'strengthen' the tampering algorithm Tam who, now, receives the 'original' sample x_i before substituting it with something else. Namely, we would first sample $x_i \leftarrow (X_i \mid y_{\leq i-1})$, and then with probability $1-p$ we let $y_i = x_i$ and with probability p we let $y_i = \mathsf{Tam}(y_{\leq i-1}, x_i)$. This definition is natural for scenarios in which the adversary gets to see the first initial sample and then might decide to change or not change it. However, as long as either (1) tampering is allowed to be inefficient or (2) $\overline{X}$ is efficiently online samplable, the power of tampering attacks under this alternative definition is the same as those under Definition 14. To see why, first note that $\mathsf{Tam}(y_{\leq i-1}, x_i)$ can always ignore the extra input x_i. In the other direction, suppose Tam' is a tampering algorithm under the alternative definition and suppose a tampering algorithm $\mathsf{Tam}(y_{\leq i-1})$ is only given $y_{\leq i-1}$. If Tam can obtain a sample $x'_i \leftarrow (X_i \mid y_{\leq i-1})$, then it could also emulate $\mathsf{Tam}'(y_{\leq i-1}, x'_i)$. Interestingly, although x_i and x'_i might be different samples, this emulation of $\mathsf{Tam}'(y_{\leq i-1}, x'_i)$ by Tam leads to the same final distribution.

Now we define what it means for a tampering adversary to successfully bias the output of a function, while being robust to changes in probabilities.

Definition 15 (Robust p-tampering attacks for biasing real functions). *Let $\overline{X} = (X_1, \ldots, X_n)$ be a joint distribution, $f\colon \mathrm{Supp}(\overline{X}) \mapsto \mathbb{R}$ a real function and Tam a tampering algorithm for $\overline{X}$.*

- *For a probability tree ρ over $\overline{X}$, we say that Tam is a ρ-tampering attack biasing $f(\overline{X})$ by at least δ, if Tam generates $\overline{Y}$ from $\overline{X}$ through a ρ-tampering attack and $\mathbb{E}[f(\overline{Y})] \geq \mathbb{E}[f(\overline{X})] + \delta$.*

– *For $p \in [0,1]$, we say that* Tam *is a p-tampering attack biasing $f(\overline{X})$ by at least δ, if* Tam *a ρ-tampering attack biasing $f(\overline{X})$ by at least δ for the constant probability tree $\rho[x_{\leq i}] = p$.*
– *We say that* Tam *is a* robust *p-tampering attack biasing $f(\overline{X})$ by at least δ, if for* every *$[p,1]$-probability tree ρ over $\overline{X}$ it holds that* Tam *is a ρ-tampering attack biasing $f(\overline{X})$ by at least δ.*

3.1 Main Results: Blockwise p-Tampering of Bounded Functions

Now, we are ready our main results that are about biasing real functions through *efficient* blockwise p-tampering attacks. We will then describe our results about the computationally unbounded setting where the tampering algorithm Tam is not necessarily polynomial time. Our main motivation for studying the computationally unbounded setting is to understand the *limitations* of what amount of bias could be achieved. We will then describe the applications of our results for attacking candidate randomness extractors (over multiple sources or variations of SV sources) through p tampering attacks.

Theorem 4 (Efficient blockwise p-tampering of bounded real functions). *Let $\overline{X} = (X_1, \ldots, X_n)$ be a joint distribution, $f\colon \mathrm{Supp}(\overline{X}) \mapsto [-1,+1]$ be a real-output function defined over $\mathrm{Supp}(\overline{X})$. Then there is a tampering algorithm Tam for $\overline{X}$ such that:*

1. **(Bias)** Tam *is a robust p-tampering attack biasing $f(\overline{X})$ by at least $\frac{p}{3+4p} \cdot \mathrm{Var}[f(\overline{X})]$. Furthermore, if the function $f\colon \mathrm{Supp}(\overline{X}) \mapsto \{-1,+1\}$ is Boolean, then the bias is at least $\frac{p}{2+2p} \cdot \mathrm{Var}[f(\overline{X})]$.*
2. **(Efficiency)** *Moreover,* Tam *could be implemented efficiently given oracle access to any online sampler $S(\cdot)$ for $\overline{X}$ and $f(\cdot)$. In particular, given only* two *samples $y_i^1, y_i^2 \leftarrow S(y_{\leq i-1})$,* $\mathsf{Tam}(y_{\leq i-1})$ *chooses between y_i^1, y_i^2 by making use of a biased coin that only depends on $\hat{f}[y_{\leq i-1}, y_i^1]$. Such biased coin could be sampled efficiently using further calls to $S(\cdot)$ and one call to $f(\cdot)$.*

See Sect. 5 (in particular Sect. 5.1) for the full proof of Theorem 4.

Theorem 4 above extends the previous result of [1] from bitwise to blockwise p-tampering. We also get bias $\Omega(p)$ though with worse constants. Also, for the case of Boolean functions, we again extend the previous result of [1] from bitwise p-tampering to blockwise p-tampering.

Importance of the efficiency features of the attacker in Theorem 4. As we will see in Theorem 5 below, we can get better biasing bounds for the Boolean case than $p \cdot \mathrm{Var}[f(\overline{X})]/4$, however, the reason that we pointed this out in Theorem 4 was that result comes along with the efficiency feature specified in Theorem 4 (and this is not the case in our Theorem 5 below). As mentioned, the attacker of Theorem 4 only needs *two honestly* generated samples $\{y_i^1, y_i^2\}$ for the next tampered block X_i and chooses one of them. Interestingly, this means that if the tampering algorithm is actually given an 'initial true value' x_i for block X_i (e.g., the honestly generated randomness to be used in a randomized

algorithm) then the tampering algorithm could basically just either keep x_i or substitute it with another fresh sample from X_i. This is a natural attack strategy when the adversary can "reset" the sampling procedure for the block X_i.

Biasing Martingales. An interesting special case of Theorem 4 is when the joint distribution $\overline{X} = (X_1, \ldots, X_n)$ is a martingale (i.e., $X_i \in \mathbb{R}$ and $\mathbb{E}[X_i \mid x_{\leq i-1}] = x_{i-1}$) and $f(\overline{X}) = X_n \in [-1, +1]$. In this case, it holds that $\hat{f}[x_{\leq i}] = x_i$, and so our attacker of Theorem 4 becomes extremely simple: given any two samples $y_i^1, y_i^2 \leftarrow (X_i \mid y_{\leq i-1})$, $\mathsf{Tam}(y_{\leq i-1})$ chooses $y_i = y_i^1$ with a probability that only depends on y_i^1 and chooses $y_i = y_i^2$ otherwise. Note that *no* further calls to the online sampler nor $f(\cdot)$ is needed! Moreover, this simple attack not only biases the final value $X_n = f(\overline{X})$ but it does bias *every* other X_i as well. The reason is that if we define $f_i(X_{\leq i}) = X_i \in [-1, +1]$, then the attacker's algorithm would be identical for biasing $f_i(\cdot)$ compared to biasing $f_n(\cdot) = f(\cdot)$. Therefore, our attack generates a p-tampering variation $\overline{Y}$ of $\overline{X}$ that *simultaneously* achieves bias $Y_i \geq X_i + (p/7) \cdot \mathrm{Var}[X_i]$ for *every* block $i \in [n]$. Moreover, the p-tampering is efficient if the martingale is online samplable.

Tampering with only a part of randomness. The specific way that the attacker of Theorem 4 chooses between the two samples $\{y_i^1, y_i^2\}$ for block X_i allows us to generalize the attack to settings where the tamping happens only over *part* of the randomness and some subsequent randomness R is also used for computing f. As we will see, this corollary would also be useful for attacking randomized learners through the so called 'targeted poisoning' attacks.

Corollary 2 (Biasing bounded 'randomized' functions). *Let* $\overline{X} = (X_1, \ldots, X_n)$ *be a joint distribution,* R *another distribution, and* $f\colon \mathrm{Supp}(\overline{X} \times R) \mapsto [-1, +1]$. *For any fixed* $x \leftarrow \overline{X}$, *let* $g(x) = \mathbb{E}_{r \leftarrow R}[f(x, r)] \in [-1, +1]$. *Then there is a tampering algorithm* Tam *for* $\overline{X}$ *(not receiving* R*) such that:*

1. **(Bias)** Tam *is a robust* p*-tampering attack biasing* $g(\overline{X})$ *by* $\geq \frac{p}{3+4p} \cdot \mathrm{Var}[g(\overline{X})]$.
2. **(Efficiency)** Tam *could be implemented efficiently given oracle access to any online sampler* $S(\cdot)$ *for* $\overline{X}$ *and* $f(\cdot, \cdot)$. *In particular,* $\mathsf{Tam}(y_{\leq i-1})$ *again chooses between two samples* $y_i^1, y_i^2 \leftarrow S(y_{\leq i-1})$ *using further calls to* $S(\cdot)$ *and one call to* $f(\cdot, \cdot)$ *and one sample from* R.

Proof (Proof of Corollary 2 using Theorem 4). To derive Corollary 2 from Theorem 4 we apply Theorem 4 directly to the function $g(x) = \mathbb{E} f(x, R)$, and we rely on the properties specified in the efficiency part of Theorem 4 to derive the efficiency of the new attacker. All we need is to provide a sample from the distribution Z (for choosing between $y_i^1, y_i^2 \leftarrow S(y_{\leq i-1})$) when we try to bias g. In order to do so, we can first sample $x \leftarrow (\overline{X} \mid y_{\leq i-1}, y_i^1)$ using $S(\cdot)$, and then output $Z \leftarrow f(x, R)$ using one sample $r \leftarrow R$. By the linearity of expectation, even though we did not really compute $g(x)$, this way of sampling Z using only one $r \leftarrow R$ has the needed properties for the (average) function g as well.

The following theorem gives a better biasing bound for the important special case of Boolean functions. On the down side, the attacker will be less efficient and asks more queries to the online sampler $S(\cdot)$.[5]

Theorem 5 (Biasing Attacks on Boolean functions). *Let* $\overline{X} = (X_1, \ldots, X_n)$ *be a joint distribution,* $f\colon \mathrm{Supp}(\overline{X}) \mapsto \{+1, -1\}$ *a Boolean function defined over* $\mathrm{Supp}(\overline{X})$*, and* $\mu = \mathbb{E}[f(\overline{X})]$*. Suppose* S *is a sampler for* $\overline{X}$ *and let* N *be an upper bound on the total binary length of* $\overline{X} = (X_1, \ldots, X_n) \in \{0,1\}^N$*, and* $\varepsilon < 1$ *be an input parameter. Then there is a tampering algorithm* Tam *for* $\overline{X}$ *that:*

1. **(Bias)** Tam *is a robust* p*-tampering attack biasing* $f(\overline{X})$ *by* $\geq \frac{p(1-\mu^2)}{2-p(1-\mu)} - \frac{\varepsilon}{1+\mu}$.[6]
2. **(Efficiency)** *Moreover,* Tam *could be implemented in time* $\mathrm{poly}(N/\varepsilon)$ *given oracle access to any online sampler* $S(\cdot)$ *for* $\overline{X}$ *and* $f(\cdot)$*. Thus, if* $\varepsilon \geq 1/\mathrm{poly}(N)$*,* $\overline{X}$ *is efficiently online samplable, and* f *is efficient, then* Tam *would be efficient as well.*

We prove our Theorem 5 using ideas from the attack of [1] also for the Boolean case. In a nutshell, we follow the same 'greedy' approach, but the analysis of the attack in the blockwise setting becomes more challenging and we can no longer get the same bias of $+p$ in the balanced case. Indeed, achieving the bias of $+p$ for balanced functions in the blockwise setting is *not* possible in general! For full proof of Theorem 5 please see the full version.

Remark 3 (Robustness vs. p-obliviousness). Note that in both Theorems 5 and 4 the attackers are robust in the sense that they work simultaneously for all $[p, 1]$ probability trees (i.e., they only rely on the lower-bound p for the probability of the tampering to happen for each block). However, this feature of the attacker should not be confused with another aspect of our attackers that they are p-*oblivious*, meaning the tampering algorithm Tam does not rely on knowing p either. Putting these two together, it means that the attackers of Theorems 4 and 5 could be "generated" independently of the probability tree ρ under which the tampering to the randomness will eventually happen, and yet the quality of obtained bias only depend on the minimum over all the probabilities under which the blocks become tamperable.

[5] The sample complexity measure is an important factor in some of the applications of our biasing attacks. For example, to attack the soundness of learning algorithms through targeted poisoning attacks, the sample complexity of the attacker translates into how much 'fresh' data is needed to substitute the original training examples when the tampering happens.

[6] The analysis of the greedy attack of [1] shows that the amount of bias is at least $p \cdot (1 - |\mu|)$. Our bound depends on $1 - \mu^2$ instead of $1 - |\mu|$. The reason behind this is that we use a better approximation of the probabilities for the output to be -1 or $+1$.

Computationally Unbounded p-Tampering. One might wonder what are the 'potential' and 'limitations' of the power of blockwise p-tampering attacks. Even though our focus in this work is on the computationally bounded setting, we also study the power and limitations of computationally unbounded p-tampering attacks. Showing the power of attackers in the unbounded model might eventually shed light into how to get better efficient attackers as well, and proving limitations in this model imply strong limits for efficient tampering algorithms as well. In Full version of this paper we show that the better biasing bound of Theorem 5 could be obtained for bounded real functions as well, but this comes with an inefficient p-tampering, and achieving this bound efficiently remains as an open question. Perhaps surprisingly, we also show that there are balanced functions over block sources where the best bias by (even inefficient) p-tampering attacks is smaller than $0.7p$. This comes in contrast with the bitwise p-tampering model where p is the optimal possible bias in general. See Full version for more details.

4 Applications of p-Tampering Biasing Attacks

In this subsection we describe some of the applications of our main results on blockwise p-tampering of bounded functions in several different contexts.

4.1 Efficient p-Tampering Attacks on Extractors

Rather than proving Theorem 2, here we prove a more general result by defining yet another generalization of SV sources based on the notion of max-divergence [23] (see Definition 6) which is tightly related to p-tampering variations. Intuitively, we will show that $\overline{X}$ is an (ℓ, γ) block SV source if the uniform distribution U_ℓ^n is a p-tampering variation of $\overline{X}$ for $p \approx \gamma$. We will then show that our p-tampering attacker of Theorem 4 produces $\overline{Y}$ such that $\overline{X}$ itself is a $O(p)$-tampering variation of $\overline{Y}$! We first define the following generalization of block-SV sources based on max-divergence.

Definition 16 (MD and MMD Sources). *Let $\overline{X} = (X_1, \ldots, X_n)$ be a joint distribution. For real number $r \geq 0$, we call a joint distribution $\overline{Y} = (Y_1, \ldots, Y_n)$ an $(\overline{X}, r)$-max-divergence (MD) source if* $\text{Supp}(\overline{Y}) = \text{Supp}(\overline{X})$ *and for all $i \in [n], x_{<i} \in \text{ValPref}(\overline{X})$ the max-divergence $D_\infty((X_i \mid x_{<i})||(Y_i \mid x_{<i}))$ is at most r. We call $\overline{Y}$ an $(\overline{X}, r)$* mutual *MD (MMD) source if in addition $\overline{X}$ is an $(\overline{Y}, r)$ MD source as well.*

Remark 4 (Sources based on other distance measures). The above definition uses max-divergence in order to limit how 'far' the source $\overline{Y}$ can be from the 'central' random process $\overline{X} = (X_1, \ldots, X_n)$. Alternative definitions could be obtained by using other distance metrics and measures. For example, we can also define $(\overline{X}, r)$ KL sources to include all distributions $\overline{Y}$ such that $D_{\text{KL}}((X_i \mid x_{<i})||(Y_i \mid x_{<i})) \leq r$. A result of [23] (see Part 3 of Lemma 3) shows that any $(\overline{X}, r)$ *mutual* MD source is also a $(\overline{X}, r')$ KL-source for $r' = r(2^r - 1)$ which is $r' \leq r^2$ for any $r \leq 1$.

The following claim shows that MD sources and p-tampering variations are tightly related. The proof directly follows from definitions of MD sources and p-variations.

Claim 1 (MD sources vs. tampering variations). *$\overline{Y} = (Y_1, \ldots, Y_n)$ is an $(\overline{X}, r)$-MD source iff it is a p-tampering variation of $\overline{X}$ for $p = 1 - 2^{-r}$.*

The following claim shows that MD sources are also related to SV block sources (in the 'reverse' direction), and its proof directly follows from the definition of MD sources and Part 2 of Lemma 3.

Claim 2 (MD sources vs. block SV sources). *For a joint distribution $\overline{X} = (X_1, \ldots, X_n)$, U_ℓ^n is an $(\overline{X}, r)$-MD source iff $\overline{X}$ is an $(\ell, \ell - r)$ block SV source. In particular, if $\overline{X}$ is an $(U_\ell^n, \ell - r)$-MMD source, then it is also an $(\ell, \ell - r)$-block SV source.*

Theorem 2 follows from Claim 2 above and the following general result about the impossibility of deterministic extraction from MMD sources.

Theorem 6 (Impossibility of extractors from MMD sources). *Let $\overline{X} = (X_1, \ldots, X_n)$ be a joint distribution with an efficient online sampler, and let $f\colon \mathrm{Supp}(\overline{X}) \mapsto \{+1, -1\}$ be an efficient Boolean function. Then, there is a p-tampering variation $\overline{Y}$ of $\overline{X}$ where:*

1. *$\overline{Y}$ is an $(\overline{X}, p)$ MMD source.*
2. *$|\mathbb{E}[f(\overline{Y})]| \geq \Omega(p)$.*
3. *$\overline{Y}$ is generated by an* efficient *tampering algorithm* Tam.

The first two items in Theorem 6 imply that f cannot be an extractor for $(\overline{X}, p)$ MMD sources for *any* $\overline{X} = (X_1, \ldots, X_n)$. Moreover, one can show that the source $\overline{Y}$ is also a $(\overline{X}, p^2)$ KL source because it is a $(\overline{X}, p)$ *mutual* MD source (see Remark 4).

Efficiency of the attacker. The last condition shows that the p-tampering attack against such f (as a candidate extractor) could be implemented by an *efficient* p-tampering attacker. We emphasize that the efficiency condition again is crucial here. In fact, if we change the statement of Theorem 6 by (1) restricting $\overline{X} = (Z \times \cdots \times Z)$ to iid distributions and more importantly (2) allowing Tam to be computationally unbounded, then we can derive this weaker version of Theorem 6 from the recent impossibility result of [6] for generalized SV sources as follows. Beigi et al. [6] showed that bit extraction with $o(1)$ bias from generalized SV sources (Definition 9) is impossible if (1) all the distributions $D \in \mathcal{D}$ available to the adversary have full support over the alphabet set C and that (2) the span of distributions $\mathcal{D}$ (see Definition 3) has full dimension $|C|$. To apply their result to MMD sources, we observe that (1) the distribution of Y_i where $\overline{Y} = (Y_1, \ldots, Y_n)$ is an $(\overline{X}, r)$ MMD source has full support (i.e., $\mathrm{Supp}(Z) = C$) and that (2) conditioned on any $y_{\leq i-1}$, the set of all possible distributions for Y_i forms a polytope with *full* rank $|\mathrm{Supp}(Z)|$.

Proof (Proof of Theorem 6). To prove Theorem 6 we use Theorem 4 and rely on some specific properties of the p-tampering attacker there. Even though the function f is Boolean, for some minor technical reasons, we will actually use the p-tampering attacker of Theorem 4 for *real* output functions. In the following we will show that this attacker has the properties listed in Theorem 6.

First note that without loss of generality, we can assume that $\mathbb{E}[f(\overline{X})] \geq 0$ (as otherwise we can work with $-f$ and bias it towards $+1$). In that case, the second and third properties of Theorem 6 follow from the main properties of Tam as stated in Theorem 4. However, for getting the first property (that it gives us an MMD source) we need to get into the actual attack's description from the proof of Theorem 4 given in Subsect. 5.1, which we also describe here. This attacker Tam (for the *real* output case) is based on one-rejection sampling (of Construction 1) modified as follows. Whenever the tampering algorithm is given the chance to tamper with a new block (which happens with probability p), the attacker itself tosses a coin and decides *not* to tamper with the block with probability 0.5, and otherwise will actually run the one-rejection sampling of Construction 1. Thus, during the execution of the p-tampering attack, the tampering actually happens with probability $p/2$.

As described above, the tampering happens with probability $p/2$, so by Claim 1, it holds that $\overline{Y}$ is an $(\overline{X}, r)$ MD source for $r \leq \lg(1/(1-p/2)) \leq p$ (by $p \in [0,1]$). On the other hand, the one-rejection sampling is actually used only with probability $p/2$. Therefore, for every possible $y_{\leq i}$, if we let $\alpha = \Pr[X_i = y_i \mid y_{\leq i-1}]$, then it holds that $\Pr[Y_i = y_i \mid y_{\leq i-1}] \leq (1-p/2)\cdot\alpha + (p/2)\cdot(2\alpha) \leq (1+p/2)\cdot\alpha$, because, either no tampering happens with probability $1-p/2$ and even if it happens, because the tampering algorithm only uses two samples for the tampered block, by a union bound, the probability of sampling y_i in this case is at most 2α, which means that $\overline{X}$ is an $(\overline{Y}, r)$ MD source for $r \leq \lg(1+p/2) \leq p$ (by $p \in [0,1]$).

Putting things together, it holds that $\overline{Y}$ is indeed an $(\overline{X}, p)$ MMD source.

4.2 Targeted Poisoning Attacks on Learners

Terminology. Let $\mathcal{D}$ be the domain containing all the objects of interest in a learning problem, and let $\mathcal{C}$ be a class of *concept* functions mapping objects in $\mathcal{D}$ to a set of labels $\mathcal{T}$. A labeled example from the set $\mathcal{D}$ for a concept function $c \in \mathcal{C}$ is a pair $x = (d, c(d))$ where $d \in \mathcal{D}$. We use $\mathcal{P}_c = \{(d, c(d)) \mid d \in \mathcal{D}\}$ to denote all the labeled examples from $\mathcal{D}$. The goal of a learning algorithm L is to produce a *hypothesis* $h \in \mathcal{H}$ after receiving a sequence $x = (x_1, \ldots, x_n)$ of labeled examples that we call the training sequence, such that h can predict the label of a given input from $\mathcal{D}$. The examples in the training sequence are usually sampled independently from a distribution X over $\mathcal{P}_c$ through an oracle $O_X(.)$ that we call the *training oracle*. A subset $\mathcal{X} \subseteq \mathcal{P}_c$ is a *test set* if we use it to evaluate the performance of the hypothesis h.

Definition 17 (Cost and average cost). *A* cost function $\mathsf{cost} : \mathcal{H} \times 2^{\mathcal{P}_c} \to [0,1]$ *captures the quality of a hypothesis, and the lower the value of* $\mathsf{cost}(h, \mathcal{X})$,

the better h is performing on the examples in $\mathcal{X}$. We define the average cost function for a learning algorithm L and a test set $\mathcal{X}$ according to a specific training oracle as follows:

$$\overline{\mathsf{cost}}_L^O(\mathcal{X}) = \mathop{\mathbb{E}}_{\substack{x_1,\ldots,x_n \leftarrow O \\ h \leftarrow L(x_1,\ldots,x_n)}} [\mathsf{cost}(h, \mathcal{X})]$$

For example the cost functions might be the fraction of examples in $\mathcal{X}$ that h generate a wrong label for. The test set itself can consist of only one point, or it might be very large to model the scenario where sampling an example from $\mathcal{X}$ is equivalent to sampling from X.[7]

Definition 18 (p-tampering training oracles). *Let O_X be the training oracle for a distribution X. A p-tampering oracle $\widehat{O}_X^p$ works as follows. Whenever the training algorithm queries this oracle, with probability $1-p$ the answer is generated from the original oracle O_X and with probability p a stateful adversary gets the control over the oracle and answers with an arbitrary pair (d,t) such that $(d,t) \in \mathcal{P}_c$. We call $\widehat{O}_X^p$ efficient, if the pair (d,t) is generated using an efficient p-tampering algorithm that takes as input 1^N, where N is the total length of the training sequence x, and all the previous samples in the training sequence.*

We can use our Theorem 4 to increase the average cost of even randomized learners where the cost could also be a real number. In the following theorem we do exactly that. However, the quality of this attack depends on the variance of the learner's success probability (as defined in Theorem 7). Thus, a provable randomized remedy against our attacks need, as the first step, to bound the variance parameter defined in Theorem 7.

Theorem 7 (Power of targeted poisoning attack against real cost functions). *Let $\mathcal{C}$ be a concept class defined over domain $\mathcal{D}$. Also let L be a (potentially randomized) learning algorithm for $\mathcal{C}$ which takes a sequence of labeled examples $x = (x_1, \ldots, x_n)$ that are sampled using an efficient training oracle O_X and outputs a hypothesis $h \in \mathcal{H}$. For any such learning algorithm L that tries to learn a concept $c \in \mathcal{C}$, any $p \in [0,1]$, any test set $\mathcal{X}$ and any cost function $\mathsf{cost} : \mathcal{H} \times 2^{\mathcal{P}_c} \to [0,1]$ there exists a p-tampering training oracle $\widehat{O}_X^p$ such that if we sample x using $\widehat{O}_X^p$ instead of O_X the average cost increases as follows:*

$$\overline{\mathsf{cost}}_L^{\widehat{O}_X^p}(\mathcal{X}) \geq \overline{\mathsf{cost}}_L^{O_X}(\mathcal{X}) + \Omega(p \cdot \sigma^2)$$

where

$$\sigma^2 = \mathrm{Var}_{x_1,\ldots,x_n \leftarrow O_X} \left[\mathop{\mathbb{E}}_{h \leftarrow L(x_1,\ldots,x_n)} [\mathsf{cost}(h, \mathcal{X})] \right].$$

[7] In case the test data comes from X itself (i.e., $\mathcal{X} \equiv X$), the average cost becomes tightly related to PAC learnability [41]. In particular, if we define cost to be one whenever the hypothesis h generates a wrong label, then any (ε, δ)-PAC learner has average cost at most $\varepsilon + \delta$. Conversely, if the average cost is at most γ, then by an averaging argument we get a $(\sqrt{\gamma}, \sqrt{\gamma})$-PAC learner.

Moreover, if L is efficient, X is efficiently samplable, and $\mathsf{cost}(\cdot)$ is efficiently computable, then the corresponding p-tampering attack is efficient as well.

Proof. Assume L uses its own randomness $r \leftarrow R$ in addition to $(x_1, \ldots, x_n)$ and outputs a hypothesis h. For a fixed test set $\mathcal{X}$, we define a function $f : \mathcal{C}_p^n \times \mathrm{Supp}(R) \to [-1, +1]$ as follows:

$$f(x_1, \ldots, x_n, r) = 2 \cdot \mathsf{cost}(L(x_1, \ldots, x_n, r), \mathcal{X}) - 1.$$

The output of the cost function is between 0 and 1, so the output of f is between -1 and $+1$. Now by using our biasing attacks over *part* of the randomness of randomized functions (i.e., Corollary 2) there exists a p-tampering variation $\overline{Y}$ of X^n, generated through an efficient tampering attack, that biases f as follows:

$$\widehat{\mu} = \mathop{\mathbb{E}}_{\substack{x_1,\ldots,x_n \leftarrow \overline{Y} \\ r \leftarrow R}} [f(x_1, \ldots, x_n, r)] > \mu + \frac{p}{7} \cdot v$$

$$\text{where} \quad \mu = \mathop{\mathbb{E}}_{\substack{x_1,\ldots,x_n \leftarrow X^n \\ r \leftarrow R}} [f(x_1, \ldots, x_n, r)]$$

$$\text{and} \quad v = \mathrm{Var}_{x_1,\ldots,x_n \leftarrow X^n} \left[\mathbb{E}_{r \leftarrow R}[f(x_1, \ldots, x_n, r)] \right].$$

Since $\overline{Y}$ is a p-tampering variation of X^n generated by an efficient tampering attack, there is an efficient p-tampering training oracle $\widehat{O}_X^p$ that generates $\overline{Y}$. By the linearity of expectation, we have $\widehat{\mu} = 2 \cdot \overline{\mathsf{cost}}_L^{\widehat{O}_X^p}(\mathcal{X}) - 1$, $\mu = 2 \cdot \overline{\mathsf{cost}}_L^{O_X^p}(\mathcal{X}) - 1$. In addition, it holds that $v = 4 \cdot \sigma^2$, so by replacing $\widehat{\mu}$, μ and v we get

$$\overline{\mathsf{cost}}_L^{\widehat{O}_X^p}(\mathcal{X}) \geq \overline{\mathsf{cost}}_L^{\widehat{O}_X^p}(\mathcal{X}) + \frac{2p}{7} \cdot \sigma^2.$$

This bound of the above theorem could be indeed very weak as it depends on the variance of the cost of the generated hypothesis. In particular, the change could be $o(1)$. As we will see, for the special case of Boolean cost functions (e.g., classification) we can increase the error arbitrarily close to one.

Theorem 8. (Power of targeted poisoning attacks against classifiers). *Let $\mathcal{C}$ be a concept class defined over domain $\mathcal{D}$. Also let L be a* deterministic*, learning algorithm for $\mathcal{C}$ which takes a sequence of labeled examples $x = (x_1, \ldots, x_n)$ that are sampled using an efficient training oracle O_X and outputs a hypothesis $h \in \mathcal{H}$. For any such learning algorithm L that tries to learn a concept $c \in \mathcal{C}$, any $p \in [0, 1]$, any test set $\mathcal{X}$ and any cost function $\mathsf{cost} : \mathcal{H} \times 2^{\mathcal{P}_c} \to \{0, 1\}$ there exist a p-tampering training oracle $\widehat{O}_X^p$ such that if we sample x using $\widehat{O}_X^p$ instead of O_X, the average cost increases as:*

$$\overline{\mathsf{cost}}_L^{\widehat{O}_X^p}(\mathcal{X}) \geq \delta + \frac{p(\delta - \delta^2)}{1 - p(1 - \delta)} \quad \textit{where} \quad \delta = \overline{\mathsf{cost}}_L^{O_X}(\mathcal{X}).$$

Moreover, if L and $\mathsf{cost}(\cdot)$ are efficient and X is efficiently samplable, then for any $\varepsilon > 0$ our p-tampering training oracle can be implemented in time $\mathrm{poly}(\frac{n}{\varepsilon \cdot \delta})$ *and achieve* $\overline{\mathsf{cost}}_L^{\widehat{O}_X^p}(\mathcal{X}) \geq \delta + \frac{p(\delta - \delta^2)}{1 - p(1 - \delta)} - \varepsilon$.

The proof of Theorem 8 is based on Theorem 5.

Proof (Proof of Theorem 8). We define a function $f : \mathcal{C}_p^n \to [-1, +1]$ as follows:

$$f(x_1, \ldots, x_n) = 2 \cdot \mathsf{cost}(L(x_1, \ldots, x_n), \mathcal{X}) - 1.$$

Now using Theorem 5, there exist a p-tampering variation $\overline{Y}$ of X^n that biases f as follows:

$$\widehat{\mu} = \mathop{\mathbb{E}}_{x_1,\ldots,x_n \leftarrow \overline{Y}} \geq \mu + \frac{p \cdot (1 - \mu^2)}{2 - p(1 - \mu)} \quad \text{where} \quad \mu = \mathop{\mathbb{E}}_{x_1,\ldots,x_n \leftarrow X^n} [f(x_1, \ldots, x_n)].$$

Since $\overline{Y}$ is a p-tampering variation of X^n, there is an p-tampering training oracle $\widehat{O}_X^p$ that generates $\overline{Y}$. With a simple calculation we have $\widehat{\mu} = 2 \cdot \overline{\mathsf{cost}}_L^{\widehat{O}_X^p}(\mathcal{X}) - 1$ and $\mu = 2 \cdot \delta - 1$. By replacing $\widehat{\mu}$ and μ we get

$$\overline{\mathsf{cost}}_L^{\widehat{O}_X^p}(\mathcal{X}) \geq \delta + \frac{p \cdot (\delta - \delta^2)}{1 - p \cdot (1 - \delta)}.$$

The efficient version of our attack also directly follows from the efficient version of Theorem 5.

A natural Boolean cost function can be defined as

$$\mathsf{cost}(h, \mathcal{X}) = \begin{cases} 0 & \text{if } h(d) = t \text{ for all } (d, t) \in \mathcal{X} \\ 1 & \text{otherwise} \end{cases}$$

where the cost function outputs 0 if the hypothesis is correct on all the examples in the test set. A special interesting case is where X' contains a single element $t \leftarrow X$ sampled from X itself, but the adversary knows this test example and hopes to increase the error of classifying t.

Corollary 3 (Doubling the error). *For every deterministic learning algorithm L that outputs a hypothesis h by taking a sequence of n labeled examples generated by an oracle O_X and for every Boolean cost function $\mathsf{cost} : \mathcal{H} \times 2^{\mathcal{P}_c} \to \{0, 1\}$, there exist a p-tampering training oracle $\widehat{O}_X^p$, using $p = \frac{1}{2(1-\delta)}$, such that doubles the average cost $\delta = \overline{\mathsf{cost}}_L^{O_X}(\mathcal{X})$ into 2δ. (I.e., for small error δ, we can double it by using $p \approx 1/2$.)*

5 Efficient p-Tampering Attacks Biasing Bounded Functions

In this section we will formally prove Theorems 4. As described in Sect. 1.2, some of the ideas (and even notation) that we use here goes back to the original work of Austrin et al. [1] and here we show how to extend these arguments to the blockwise setting and overcome challenges that emerge.

Before doing so, we need to define some useful notation for the notions that naturally come up in our proofs. We will also make some basic observations about these quantities before proving our main theorems.

Definition 19 (Functions $\hat{f}, g, \mathcal{G}, \mathcal{A}, \mathcal{Q}$). *Suppose $f\colon \mathrm{Supp}(\overline{X}) \mapsto \mathbb{R}$ is defined over a joint distribution $\overline{X} = (X_1, \ldots, X_n)$, $i \in [n]$, and $x_{\leq i} \in \mathrm{ValPref}(\overline{X})$ is a valid prefix for $\overline{X}$. Then we define the following with respect to $f, \overline{X}, x_{\leq i}$.*

- *$f_{x_{\leq i}}(\cdot)$ is a function defined as $f_{x_{\leq i}}(x_{\geq i+1}) = f(x)$ where $x = (x_{\leq i}, x_{\geq i+1})$.*
- *$\hat{f}[x_{\leq i}] = \mathbb{E}_{x_{\geq i+1} \leftarrow (X_{\geq i+1} | x_{\leq i})}[f_{x_{\leq i}}(x_{\geq i+1})]$. We also use $\mu = \hat{f}[\varnothing]$ to denote $\hat{f}[x_{\leq 0}] = \mathbb{E}[f(\overline{X})]$.*
- *We define the* gain *of the "node" $x_{\leq i}$ (compared to its parent $x_{\leq i-1}$) as $g[x_{\leq i}] = \hat{f}[x_{\leq i}] - \hat{f}[x_{\leq i-1}]$. This defines the change in $\hat{f}[x_{\leq i}]$ after moving to the i'th block.*
- *For every $x_{\leq i-1}$ and every distribution Z that* could depend *on $x_{\leq i-1}$ (e.g., Z is the output of a randomized algorithm that takes $x_{\leq i-1}$ as input) and $\mathrm{Supp}(Z \mid x_{\leq i-1}) \subseteq \mathrm{Supp}(X_i \mid x_{\leq i-1})$ we define:*
  - *The* average of the gain *over the "children" of node $x_{\leq i-1}$ under distribution $(Z \mid x_{\leq i-1})$:*

$$\mathcal{G}_Z[x_{\leq i-1}] = \mathop{\mathbb{E}}_{x_i \leftarrow (Z | x_{\leq i-1})}[g[x_{\leq i}]].$$

  - *The average of the* squares *of the gains:*

$$\mathcal{Q}_Z[x_{\leq i-1}] = \mathop{\mathbb{E}}_{x_i \leftarrow (Z | x_{\leq i-1})}\left[g[x_{\leq i}]^2\right].$$

Notation. Throughout the following sections, whenever we define $\overline{X}$ and f, then we will use all the notations defined in Definition 19 with respect to f and $\overline{X}$ even if there are other distributions like $\overline{Y}$ defined.

The following lemma directly follows from the definition of μ and $g[x_{\leq i}]$.

Proposition 1. *For every $x \in \mathrm{Supp}(\overline{X})$, $f(x) = \mu + \sum_{i \in [n]} g[x_{\leq i}]$.*

The following two intuitive propositions also follow from the definition of $\mathcal{G}_{X_i}[x_{\leq i-1}]$ (See the full version for the proofs.).

Proposition 2. *For every valid prefix $x_{\leq i-1} \in \mathrm{ValPref}(\overline{X})$, we have $\mathcal{G}_{X_i}[x_{\leq i-1}] = 0$.*

Proposition 3. *Let $f\colon \mathrm{Supp}(\overline{X}) \mapsto \mathbb{R}$ be any real-output function. Then for any distribution $\overline{Y}$ such that $\mathrm{Supp}(\overline{Y}) \subseteq \mathrm{Supp}(\overline{X})$ it holds that $\mathbb{E}[f(\overline{Y})] - \mathbb{E}[f(\overline{X})] = \sum_{i \in [n]} \mathbb{E}_{Y_{\leq i-1}}\left[\mathcal{G}_{Y_i}[Y_{\leq i-1}]\right]$.*

The above proposition holds for any distribution $\overline{Y}$ as long as $\mathrm{Supp}(\overline{Y}) \subseteq \mathrm{Supp}(\overline{X})$, but the following is just about ρ-tampering variations.

Proposition 4. *For any probability tree ρ over $\overline{X}$, and any ρ-tampering variation $\overline{Y}$ of $\overline{X}$ generated by a (possibly randomized) tampering algorithm Tam, and for any $y_{\leq i-1} \in \mathrm{ValPref}(\overline{X})$, it holds that $\mathcal{G}_{Y_i}[y_{\leq i-1}] = \rho[y_{\leq i-1}] \cdot \mathcal{G}_{\mathsf{Tam}}[y_{\leq i-1}]$.*

Proof. The proof simply follows from the definition of ρ-tampering variations. When we sample from the distribution $(Y_i \mid \overline{Y}_{\leq i-1} = y_{\leq i})$, by definition, with probability $1-\rho[y_{\leq i-1}]$ we will be sampling Y_i from $(X_i \mid X_{\leq i-1} = y_{\leq i-1})$ which by Proposition 2 leads to gaining $\mathcal{G}_{X_i}[y_{\leq i-1}] = 0$, and with probability $\rho[y_{\leq i-1}]$ we will be sampling Y_i from $\mathsf{Tam}(y_{\leq i-1})$ which leads to gaining $\mathcal{G}_{\mathsf{Tam}}[y_{\leq i-1}]$. Putting together, this implies an average gain of $\rho[y_{\leq i-1}] \cdot \mathcal{G}_{\mathsf{Tam}}[y_{\leq i-1}]$.

5.1 Biasing Real-Output Functions: Proving Theorem 4

In this Section we will prove our Theorem 4.

Construction 1. *Let* $\overline{X} = (X_1, \ldots, X_n)$ *be the joint distribution and* $f\colon \mathrm{Supp}(\overline{X}) \mapsto [-1,+1]$. *The* one rejection sampling *tampering algorithm* ORSam *works as follows. Given the valid prefix* $y_{\leq i-1} \in \mathrm{ValPref}(\overline{X})$, *the tampering algorithm would sample* $y_{\geq i} \leftarrow (X_{\geq i} \mid y_{\leq i-1})$ *by multiple invocations of the online sampler* S. *Then it computes* $s = f(y_1, \ldots, y_n)$ *and output from the following random variable.*

$$T = \begin{cases} \textit{Case 1: with probability } \frac{1+s}{2} \textit{ output } y_i. \\ \textit{Case 2: with probability } \frac{1-s}{2} \textit{ output a fresh sample } y'_i \leftarrow S(y_{\leq i-1}). \end{cases}$$

Claim 3. *For every* $f : \mathrm{Supp}(\overline{X}) \to [-1,+1]$ *and every* $[p,q]$*-probability tree* ρ *over* $\overline{X}$, *the tampering algorithm* ORSam *of construction 1 generates a* ρ*-tampering variation* $\overline{Y}$ *of* $\overline{X}$ *such that* $\mathbb{E}[f(\overline{Y})] \geq \mathbb{E}[f(\overline{X})] + \frac{p\cdot(1-q)}{2+2p-2q-pq} \cdot \mathrm{Var}[f(\overline{X})]$, *and if* $f : \mathrm{Supp}(\overline{X}) \to \{+1,-1\}$ *is Boolean, then* $\mathbb{E}[f(\overline{Y})] \geq \mathbb{E}[f(\overline{X})] + \frac{p}{2+2p} \cdot \mathrm{Var}[f(\overline{X})]$.

We first prove Theorem 4 using Claim 3, and then we will prove Claim 3

Proof (Proof of Theorem 4). We need to show that there is an attack that can bias f by $\Omega(p)$. For the Boolean case the proof follows directly from the statement of Claim 3. For the case of real-output functions we use an attacker that with probability 0.5 uses uses a fresh sample, and with probability 0.5 it runs the one-rejection sampling attack of Construction 1. This algorithm gives a ρ-tampering variation $\overline{Y}$ of $\overline{X}$ such that $\forall y_{\leq i} \in \mathrm{ValPref}(\overline{X})$, $\frac{p}{2} \leq \rho[y_{\leq i}] \leq \frac{1}{2}$ so using Claim 3 we have:

$$\mathbb{E}[f(\overline{Y})] - \mathbb{E}[f(\overline{X})] \geq \frac{p/4}{1+3p/4} \mathrm{Var}[f(\overline{X})] = \frac{p}{4+3p} \mathrm{Var}[f(\overline{X})].$$

In the rest of this section we will first prove three lemmas and then will use them to prove Claim 3. All along we use $\overline{Y}$ to denote the ρ-tampering variation of $\overline{X}$ generated by one rejection sampling algorithm ORSam of Construction 1.

Claim 4. *Let* $T \equiv \mathsf{ORSam}(y_{\leq i-1})$ *be a random variable defined over the randomness of* ORSam *running on a valid prefix* $y_{\leq i-1} \in \mathrm{ValPref}(\overline{X})$. *The probability distribution of this random variable is:*

$$\Pr[T = y_i] = \left(1 + \frac{g[y_{\leq i}]}{2}\right) \cdot \Pr[X_i = y_i \mid y_{\leq i-1}].$$

Proof. We have two cases in the attack. We first compute the probability of Case 1.

$$\begin{aligned}\Pr[\text{Case 1} \wedge T = y_i] &= \mathop{\mathbb{E}}_{y_{>i} \leftarrow (X_{>i} | y_{\le i-1})} \left[\frac{1+f(y)}{2}\right] \cdot \Pr[X_i = y_i \mid y_{\le i-1}] \\ &= \left(\frac{1+\hat{f}[y_{\le i}]}{2}\right) \cdot \Pr[X_i = y_i \mid y_{\le i-1}].\end{aligned}$$

On the other hand, the probability of Case 2 is

$$\begin{aligned}\Pr[\text{Case 2} \wedge T = y_i] &= \Pr[T = y_i \mid \text{Case 2}] \cdot \Pr[\text{Case 2}] \\ &= \Pr[X_i = y_i \mid y_{\le i-1}] \cdot \mathop{\mathbb{E}}_{y_{>i-1} \leftarrow (X_{>i-1} | y_{\le i-1})} \left[\frac{1-f(y)}{2}\right] \\ &= \Pr[X_i = y_i \mid y_{\le i-1}] \cdot \left(\frac{1-\hat{f}[y_{\le i-1}]}{2}\right).\end{aligned}$$

Thus, we have

$$\begin{aligned}\Pr[T = y_i] &= \Pr[\text{Case 1} \wedge T = y_i] + \Pr[\text{Case 2} \wedge T = y_i] \\ &= \left(\frac{1+\hat{f}[y_{\le i}]}{2}\right) \cdot \Pr[X_i = y_i \mid y_{\le i-1}] \\ &\quad + \Pr[X_i = y_i \mid y_{\le i-1}] \cdot \left(\frac{1-\hat{f}[y_{\le i-1}]}{2}\right) \\ &= \left(1 + \frac{g[y_{\le i}]}{2}\right) \cdot \Pr[X_i = y_i \mid y_{\le i-1}].\end{aligned}$$

Corollary 4. *For any $y_{\le i} \in \mathrm{ValPref}(\overline{X})$, it holds that*

$$\Pr[Y_i = y_i \mid y_{\le i-1}] = \left(1 + \frac{\rho[y_{\le i-1}] \cdot g[y_{\le i}]}{2}\right) \cdot \Pr[X_i = y_i \mid y_{\le i-1}].$$

Proof. By definition of $\overline{Y}$ we have

$$\begin{aligned}\Pr[Y_i = y_i \mid y_{\le i-1}] &= (1 - \rho[y_{\le i-1}]) \cdot \Pr[X_i = y_i \mid y_{\le i-1}] \\ &\quad + \rho[y_{\le i-1}] \cdot \Pr[y_i = \mathsf{ORSam}(y_{\le i-1})] \\ \text{(by Claim 4)} \quad &= (1 - \rho[y_{\le i-1}] + \rho[y_{\le i-1}] \cdot (1 + \frac{g[y_{\le i}]}{2})) \Pr[X_i = y_i \mid y_{\le i-1}] \\ &= \left(1 + \frac{\rho[y_{\le i-1}] \cdot g[y_{\le i}]}{2}\right) \cdot \Pr[X_i = y_i \mid y_{\le i-1}].\end{aligned}$$

Lemma 1. *Let $\overline{X} = (X_1, \ldots, X_n)$. For every function $f\colon \mathrm{Supp}(\overline{X}) \to [-1, +1]$ and every $[p, 1]$-probability tree ρ over $\overline{X}$, if $\overline{Y}$ is the ρ-tampering variation of*

distribution $\overline{X}$ *generated by tampering algorithm* ORSam *of construction 1, and if* $\mu = \mathbb{E}[f(\overline{X})]$*, then it holds that*

$$\mathbb{E}[f(\overline{Y})] \geq \mu + \frac{p}{2(1+p)} \cdot \left(\mathbb{E}[f(\overline{Y})^2] - \mu^2\right).$$

Before proving the above lemma, we will need to prove several other claims.

Claim 5 (One rejection sampling's local gains). *For any* $y_{\leq i} \in \mathrm{ValPref}(\overline{X})$*, it holds that*

$$\mathcal{G}_{\mathsf{ORSam}}[y_{\leq i-1}] = \mathcal{Q}_{X_i}[y_{\leq i-1}]/2.$$

Proof. First note that $\mathcal{G}_{\mathsf{ORSam}}[y_{\leq i-1}] = \sum_{y_i} \Pr[y_i = \mathsf{ORSam}(y_{\leq i})] \cdot g[y_{\leq i}]$. By Claim 4 we get

$$\begin{aligned}
\mathcal{G}_{\mathsf{ORSam}}[y_{\leq i-1}] &= \sum_{y_i} \Pr[X_i = y_i \mid y_{\leq i-1}] \cdot \left(1 + \frac{g[y_{\leq i}]}{2}\right) \cdot g[y_{\leq i}] \\
&= \sum_{y_i} \Pr[X_i = y_i \mid y_{\leq i-1}] \cdot g[y_{\leq i}] + \sum_{y_i} \Pr[X_i = y_i \mid y_{\leq i-1}] \cdot \frac{g[y_{\leq i}]^2}{2} \\
&= \mathcal{G}_{X_i}[y_{\leq i-1}] + \frac{\mathcal{Q}_{X_i}[y_{\leq i-1}]}{2}.
\end{aligned}$$

By Proposition 2 we also know that $\mathcal{G}_{X_i}[y_{\leq i-1}] = 0$, so $\mathcal{G}_{\mathsf{ORSam}}[y_{\leq i-1}] = \mathcal{Q}_{X_i}[y_{\leq i-1}]/2$.

Corollary 5. *For any* $y_{\leq i-1} \in \mathrm{ValPref}(\overline{X})$*, it holds that* $\mathcal{G}_{Y_i}[y_{\leq i-1}] = \frac{\rho[y_{\leq i-1}]}{2} \cdot \mathcal{Q}_{X_i}[y_{\leq i-1}]$.

Proof.

$$\begin{aligned}
\mathcal{G}_{Y_i}[y_{\leq i-1}] &= \sum_{y_i} \Pr[y_i = Y_i \mid y_{\leq i-1}] \cdot g[y_{\leq i}] \\
&= \sum_{y_i} \Big((1 - \rho[y_{\leq i-1}]) \cdot \Pr[y_i = X_i \mid y_{\leq i-1}]\Big) \cdot g[y_{\leq i}] \\
&+ \sum_{y_i} \Big(\rho[y_{\leq i-1}] \cdot \Pr\left[y_i = \mathsf{ORSam}(y_{\leq i-1})\right]\Big) \cdot g[y_{\leq i}] \\
&= (1 - \rho[y_{\leq i-1}]) \cdot \mathcal{G}_{X_i}[y_{\leq i-1}] + \rho[y_{\leq i-1}] \cdot \mathcal{G}_{\mathsf{ORSam}}[y_{\leq i-1}] \\
\text{(by Proposition 2)} \quad &= \rho[y_{\leq i-1}] \cdot \mathcal{G}_{\mathsf{ORSam}}[y_{\leq i-1}] \\
\text{(by Claim 5)} \quad &= \frac{\rho[y_{\leq i-1}]}{2} \cdot \mathcal{Q}_{X_i}[y_{\leq i-1}].
\end{aligned}$$

Corollary 6. $\mathbb{E}_{\overline{Y}}[f(\overline{Y})] = \mu + \sum_{i=1}^{n} \mathbb{E}_{Y_{\leq i-1}}\left[\frac{\rho[Y_{\leq i-1}]}{2} \cdot \mathcal{Q}_{X_i}[Y_{\leq i-1}]\right]$.

Proof. Using Claim 3, we have $\mathbb{E}_{\overline{Y}}[f(\overline{Y})] = \mu + \sum_{i=1}^{n} \mathbb{E}_{Y_{\leq i-1}}\left[\mathcal{G}_{Y_i}[Y_{\leq i-1}]\right]$. By also using Corollary 5 we obtain $\mathbb{E}_{\overline{Y}}[f(\overline{Y})] = \mu + \sum_{i=1}^{n} \mathbb{E}_{Y_{\leq i-1}}\left[\frac{\rho[Y_{\leq i-1}]}{2} \cdot \mathcal{Q}_{X_i}[Y_{\leq i-1}]\right]$.

Claim 6. *For every $x \in \mathrm{Supp}(\overline{X})$, it holds that*

$$f(x)^2 = \mu^2 + \sum_{i=1}^{n} \left(g[x_{\le i}]^2 + 2\hat{f}[x_{\le i-1}] \cdot g[x_{\le i}] \right).$$

Proof. By squaring the equation in Proposition 1 we get

$$f(x)^2 = \mu^2 + \sum_{i=1}^{n} g[x_{\le i}]^2 + 2\sum_{i=1}^{n} g[x_{\le i}] \cdot (\mu + \sum_{j=1}^{i-1} .g[x_{\le j}])$$

By the definition of $g[x_{\le j}]$ it holds that $\hat{f}[x_{\le i-1}] = \mu + \sum_{j=1}^{i-1} g[x_{\le j}]$. So we get

$$f(x)^2 = \mu^2 + \sum_{i=1}^{n} \left(g[x_{\le i}]^2 + 2\hat{f}[x_{\le i-1}] \cdot g[x_{\le i}] \right).$$

You can find the proof of the following two claims in the full version of this paper.

Claim 7. *For any $y_{\le i-1} \in \mathrm{ValPref}(\overline{X})$, it holds that*

$$\mathcal{Q}_{Y_i}[y_{\le i-1}] = \mathcal{Q}_{X_i}[y_{\le i-1}] + \mathop{\mathbb{E}}_{X_i | y_{\le i-1}} \left[\frac{\rho[y_{\le i-1}]}{2} \cdot g[(y_{\le i-1}, X_i)]^3 \right].$$

Claim 8. *For any $y_{\le i-1} \in \mathrm{ValPref}(\overline{X})$, it holds that*

$$\hat{f}[y_{\le i-1}] \cdot \mathcal{Q}_{X_i}[y_{\le i-1}] + \mathop{\mathbb{E}}_{X_i | y_{\le i-1}} \left[g[(y_{\le i-1}, X_i)]^3 \right] \le \mathcal{Q}_{X_i}[y_{\le i-1}].$$

Claim 9. *For any $[p,1]$-probability tree ρ over $\overline{X}$ it holds that*

$$\mathbb{E}[f(\overline{Y})^2] \le \mu^2 + \frac{1+p}{p} \cdot \sum_{i=1}^{n} \mathop{\mathbb{E}}_{Y_{\le i-1}} [\rho[Y_{\le i-1}] \mathcal{Q}_{X_i}[Y_{\le i-1}]].$$

Proof. Using Claim 6 we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{\overline{Y}}[f(\overline{Y})^2] - \mu^2 &= \sum_{i=1}^{n} \mathop{\mathbb{E}}_{\overline{Y}} \left[g[Y_{\le i}]^2 + 2\hat{f}[Y_{\le i-1}] \cdot g[Y_{\le i}] \right] \\
&= \sum_{i=1}^{n} \mathop{\mathbb{E}}_{Y_{\le i-1}} [\mathop{\mathbb{E}}_{Y_i | Y_{\le i-1}} [g[Y_{\le i}]^2]] + 2\sum_{i=1}^{n} \mathop{\mathbb{E}}_{Y_{\le i-1}} [\hat{f}[Y_{\le i-1}] \cdot \mathop{\mathbb{E}}_{Y_i | Y_{\le i-1}} [g[Y_{\le i}]]] \\
&= \sum_{i=1}^{n} \mathop{\mathbb{E}}_{Y_{\le i-1}} [\mathcal{Q}_{Y_i}[Y_{\le i-1}]] + 2\sum_{i=1}^{n} \mathop{\mathbb{E}}_{Y_{\le i-1}} [\hat{f}[Y_{\le i-1}] \cdot \mathcal{G}_{Y_i}[\overline{Y}_{\le i-1}]] \\
\text{(by Claim 7)} \qquad &= \sum_{i=1}^{n} \mathop{\mathbb{E}}_{Y_{\le i-1}} \left[\mathcal{Q}_{X_i}[Y_{\le i-1}] + \frac{\rho[Y_{\le i-1}]}{2} \mathop{\mathbb{E}}_{X_i | Y_{\le i-1}} [g[(Y_{\le i-1}, X_i)]^3] \right]
\end{aligned}$$

$$
\begin{aligned}
&+\sum_{i=1}^{n} \mathop{\mathbb{E}}_{Y_{\leq i-1}} \left[2\hat{f}[Y_{\leq i-1}] \cdot \mathcal{G}_{Y_i}[Y_{\leq i-1}]\right] \\
\text{(by Corollary 5)} \quad &= \sum_{i=1}^{n} \mathop{\mathbb{E}}_{Y_{\leq i-1}} \left[\mathcal{Q}_{X_i}[Y_{\leq i-1}] + \frac{\rho[Y_{\leq i-1}]}{2} \mathop{\mathbb{E}}_{X_i|Y_{\leq i-1}} [g[(Y_{\leq i-1}, X_i)]^3]\right] \\
&+\sum_{i=1}^{n} \mathop{\mathbb{E}}_{Y_{\leq i-1}} \left[\rho[Y_{\leq i-1}]\hat{f}[Y_{\leq i-1}]\mathcal{Q}_{X_i}[Y_{\leq i-1}]\right] \\
\text{(by Claim 8)} \quad &\leq \sum_{i=1}^{n} \mathop{\mathbb{E}}_{Y_{\leq i-1}} \left[\mathcal{Q}_{X_i}[Y_{\leq i-1}] + \frac{\rho[Y_{\leq i-1}]}{2} \cdot \mathcal{Q}_{X_i}[Y_{\leq i-1}]\right] \\
&+\sum_{i=1}^{n} \mathop{\mathbb{E}}_{Y_{\leq i-1}} \left[\frac{\rho[Y_{\leq i-1}]}{2} \cdot \hat{f}[Y_{\leq i-1}] \cdot \mathcal{Q}_{X_i}[Y_{\leq i-1}]\right] \\
\text{(by } \hat{f}[Y_{\leq i-1}] \leq 1) \quad &\leq \sum_{i=1}^{n} \mathop{\mathbb{E}}_{Y_{\leq i-1}} [(1 + \rho[Y_{\leq i-1}]) \cdot \mathcal{Q}_{X_i}[Y_{\leq i-1}]] \\
\text{(by } \rho[Y_{\leq i-1}] \geq p) \quad &\leq \left(\frac{1}{p} + 1\right) \cdot \sum_{i=1}^{n} \mathop{\mathbb{E}}_{Y_{\leq i-1}} [\rho[Y_{\leq i-1}] \cdot \mathcal{Q}_{X_i}[Y_{\leq i-1}]].
\end{aligned}
$$

Now we will prove Lemma 1.

Proof (Proof of Lemma 1). Using Claim 9 we have

$$\sum_{i=1}^{n} \mathop{\mathbb{E}}_{Y_{\leq i-1}} [\rho[Y_{\leq i-1}]\mathcal{Q}_{X_i}[Y_{\leq i-1}]] \geq \frac{p}{1+p} \cdot (\mathop{\mathbb{E}}_{\overline{Y}}[f(\overline{Y})^2] - \mu^2)$$

By also applying Corollary 6 we get $\mathbb{E}[f(\overline{Y})] \geq \mu + \frac{p}{2(1+p)} \cdot (\mathbb{E}[f(\overline{Y})^2] - \mu^2)$.

Lemma 2. *For every function $f\colon \overline{X} \to [-1,+1]$ and every $[0,q]$-probability tree ρ over $\overline{X}$, if $\overline{Y}$ is the ρ-tampering variation of distribution $\overline{X}$ generated by tampering algorithm* ORSam *of construction 1 it holds that*

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{\overline{Y}}[f(\overline{Y})] + \frac{1-q}{q} \cdot \mathop{\mathbb{E}}_{\overline{Y}}[f(\overline{Y})^2] + \frac{1-q}{2q} \cdot \mathop{\mathbb{E}}_{\overline{Y}}[f(\overline{Y})^2]^2 \geq \mathbb{E}[f(\overline{X})] + \frac{1-q}{q}\mathbb{E}[f(\overline{X})^2] \\
&\quad + \frac{1-q}{2q} \cdot \mathbb{E}[f(\overline{X})^2]^2.
\end{aligned}
$$

Before proving Lemma 2 we need to define a few useful functions.

Definition 20 (Functions $t, r, \hat{t}$ and the potential function). *Let $t\colon \mathrm{Supp}(\overline{X}) \to [0,1]$ be the square of f, namely for every $y \in \mathrm{Supp}(\overline{X}), t(y) = f(y)^2$. We also define $\hat{t}$ the same way we defined $\hat{f}$ in Definition 19. Namely, for every valid prefix $x_{\leq i} \in \mathrm{ValPref}(\overline{X})$ we have $\hat{t}[x_{\leq i}] = \mathbb{E}_{x_{\geq i+1} \leftarrow (X_{\geq i+1}|x_{\leq i})}[t_{x_{\leq i}}(x_{\geq i+1})]$. Also for every valid prefix $y_{\leq i}$ for $\overline{X}$ let r be defined as $r[y_{\leq i}] = \hat{t}[y_{\leq i}] - \hat{t}[y_{\leq i-1}]$ and for every $i \in [n]$ and every valid prefix $y_{\leq i} \in \mathrm{ValPref}(\overline{X})$ let the potential function Φ be defined as follows: $\Phi(y_{\leq i}) = \hat{f}[y_{\leq i}] + \frac{1-q}{q} \cdot \hat{t}[y_{\leq i}] + \frac{1-q}{2q} \cdot (\hat{t}[y_{\leq i}])^2$.*

Proposition 5. *If* $y_{\leq i} \in \mathrm{ValPref}(\overline{X})$*, then* $\mathbb{E}_{y_i \leftarrow (X_i | y_{\leq i-1})}[r[y_{\leq i}]] = 0$.

Proof. The proof is identical to the proof of Proposition 2.

Claim 10 (Potential function does not decrease). $\mathbb{E}[\Phi(Y_{\leq i})] \geq \mathbb{E}[\Phi(Y_{\leq i-1})]$.

Proof. Please see the full version for the proof.

Now, Lemma 2 immediately follows from Claim 10.

Proof (Proof of Lemma 2). Using Claim 10 together with a simple induction we get

$$\mathbb{E}[\Phi(Y_{\leq n})] \geq \mathbb{E}[\Phi(Y_{\leq 0})]$$

which means

$$\mathbb{E}[f(\overline{Y})] + \frac{1-q}{q} \cdot \mathbb{E}[f(\overline{Y})^2] + \frac{1-q}{2q} \cdot \mathbb{E}[f(\overline{Y})^2]^2 \geq \mathbb{E}[f(\overline{X})] + \frac{1-q}{q}\mathbb{E}[f(\overline{X})^2] + \frac{1-q}{2q} \cdot \mathbb{E}[f(\overline{X})^2]^2.$$

Finally, we prove Claim 3.

Proof (Proof of Claim 3). Let $\alpha = \mathbb{E}[f(\overline{X})^2] - \mathbb{E}[f(\overline{Y})^2]$ and $\mathrm{Var}[f(\overline{X})] = \mathbb{E}[f(\overline{X})^2] - \mathbb{E}[f(\overline{X})]^2$. Using Lemma 1 we have

$$\mathbb{E}[f(\overline{Y})] \geq \mathbb{E}[f(\overline{X})] + \frac{p}{2(1+p)} \cdot (\mathrm{Var}[f(\overline{X})] - \alpha). \tag{2}$$

If $\alpha < 0$, using this inequality we have the following. (We assume $q < 1$ otherwise the inequality below holds trivially).

$$\begin{aligned}
\mathbb{E}[f(\overline{Y})] &\geq \mathbb{E}[f(\overline{X})] + \frac{p}{2(1+p)} \cdot \mathrm{Var}[f(\overline{X})] \\
&\geq \mathbb{E}[f(\overline{X})] + \frac{p(1-q)}{2(1+p)(1-q)} \cdot \mathrm{Var}[f(\overline{X})] \\
&\geq \mathbb{E}[f(\overline{X})] + \frac{p(1-q)}{2+2\cdot p-2\cdot q-p\cdot q} \cdot \mathrm{Var}[f(\overline{X})].
\end{aligned}$$

So we can assume $\alpha \geq 0$, in which case by also using Lemma 2 we get

$$\begin{aligned}
\mathbb{E}[f(\overline{Y})] - \mathbb{E}[f(\overline{X})] &\geq \frac{(1-q)}{q} \cdot (\mathbb{E}[f(\overline{X})^2] - \mathbb{E}[f(\overline{Y})^2]) + \frac{(1-q)}{2q}(\mathbb{E}[f(\overline{X})^2]^2 \\
-\mathbb{E}[f(\overline{Y})^2]^2)\,(\text{By}\,\alpha \geq 0) &\geq \frac{(1-q)}{q} \cdot (\mathbb{E}[f(\overline{X})^2] - \mathbb{E}[f(\overline{Y})^2]) = \frac{\alpha \cdot (1-q)}{q}.
\end{aligned} \tag{3}$$

By combining the Inequalities 2 and 3 we get

$$\begin{aligned}
\mathbb{E}[f(\overline{Y})] - \mathbb{E}[f(\overline{X})] &\geq \max\left(\frac{p}{2(1+p)} \cdot (\mathrm{Var}[f(\overline{X})] - \alpha), \frac{\alpha \cdot (1-q)}{q}\right) \\
&\geq \frac{p(1-q)}{2+2\cdot p-2\cdot q-p\cdot q}\,\mathrm{Var}[f(\overline{X})].
\end{aligned}$$

where the minimum is achieved when at $\frac{p\cdot(\mathrm{Var}[f(\overline{X})]-\alpha)}{2(1+p)} = \frac{\alpha\cdot(1-q)}{q}$ at $\alpha = \frac{\mathrm{Var}[f(\overline{X})]\cdot p\cdot q}{2p+2-pq-2q}$.

Remark 5. Austrin et al. [1] analyzed their mild greedy attack using a different potential function defined as follows:

$$\Phi(y_{\le i}) = \hat{f}[y_{\le i}] + \frac{1}{2}\cdot\hat{t}[y_{\le i}] + \frac{1}{4}\cdot(\hat{t}[y_{\le i}])^2.$$

Using this potential function they show that the amount of bias for mild greedy is at least $\frac{p}{1+4p}\cdot\mathrm{Var}[f(\overline{X})]$. Using our p-dependent potential function

$$\Phi(y_{\le i}) = \hat{f}[y_{\le i}] + \frac{1}{2p}\cdot\hat{t}[y_{\le i}] + \frac{1}{4p}\cdot(\hat{t}[y_{\le i}])^2$$

one can get a slightly better bound (mainly for small p) of $\frac{p}{1+2p+2p^2}\cdot\mathrm{Var}[f(/\overline{X})]$.

6 Open Questions

We conclude by describing some open questions and interesting directions for future research.

Power of k-sampling attacks for small k. A natural yet more general class of attacks that include k-resetting attacks at special case is the class of $k+1$ sampling attacks in which the tampering algorithm first gets $k+1$ samples from the distribution of the i'th tampered block and then it chooses one of these samples (perhaps by calls to the online sampler and the function f). Our ℓ-greedy algorithm is indeed an ℓ sampling attack but to get good bias, it needs to use many $\ell = \mathrm{poly}(n/\varepsilon)$ samples. What is the power of ℓ-sampling attacks in general, when ℓ is small, e.g. constant?

Power of 'very' efficient viruses. What is the power of tampering attacks whose computational resources is not sufficient for sampling the next block or even computing f? Such tampering algorithms are natural for cryptographic attacks where computing f is heavy and the virus might prefer to use very limited resources not to be detected by the system. Our efficient tampering attacks of Theorems 4 and 5 both need to run the online sampler as well as the function f. It remains an interesting future direction to study the power of limited tampering attacks whose decisions are more 'local' and cannot be based on sampling the blocks from the original distribution or computing f.

We conjecture that such efficient viruses that cannot depend on f or the distribution $\overline{X}$ are not powerful to achieve constant bias $\Omega(p)$. However, it is interesting to find out what is the *minimum* number of calls needed to f or the sampler for getting bias $\Omega(p)$.

Biasing up vs. biasing either way. Our Theorems 4 and 5 always bias the function towards +1. Inspired by models of attacks against coin-tossing protocols

[8,14,15,17,29,34] one can ask the following questions. What is the power of p-tampering biasing attacks whose goal is to *either* bias the average of the function up *or* bias it down? Some of the applications of our biasing attacks (e.g., against learners) need to bias the function always in a fixed direction to increase the 'error', but other attacks (e.g., against extractors) could achieve their goal by biasing the function in either direction.

Acknowledgement. We thank Dimitrios Diochnos, Yevgeniy Dodis, and Yanjun Qi for useful discussions.

References

1. Austrin, P., Chung, K.-M., Mahmoody, M., Pass, R., Seth, K.: On the impossibility of cryptography with tamperable randomness. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 462–479. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44371-2_26
2. Awasthi, P., Balcan, M.F., Long, P.M.: The power of localization for efficiently learning linear separators with noise. In: Proceedings of the 46th Annual ACM Symposium on Theory of Computing, pp. 449–458. ACM (2014)
3. Azar, Y., Broder, A.Z., Karlin, A.R., Linial, N., Phillips, S.: Biased random walks. In: Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing, pp. 1–9. ACM (1992)
4. Barak, B., Halevi, S.: A model and architecture for pseudo-random generation with applications to/dev/random. In: Proceedings of the 12th ACM Conference on Computer and Communications Security, pp. 203–212. ACM (2005)
5. Barreno, M., Nelson, B., Joseph, A.D., Tygar, J.D.: The security of machine learning. Mach. Learn. **81**(2), 121–148 (2010)
6. Beigi, S., Etesami, O., Gohari, A.: Deterministic randomness extraction from generalized and distributed santha-vazirani sources. SIAM J. Comput. **46**(1), 1–36 (2017)
7. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 1–19. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44371-2_1
8. Berman, I., Haitner, I., Tentes, A.: Coin flipping of any constant bias implies one-way functions. In: Proceedings of the 46th Annual ACM Symposium on Theory of Computing, pp. 398–407. ACM (2014)
9. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997). doi:10.1007/3-540-69053-0_4
10. Bshouty, N.H., Eiron, N., Kushilevitz, E.: PAC learning with nasty noise. Theor. Comput. Sci. **288**(2), 255–275 (2002)
11. Chandran, N., Goyal, V., Mukherjee, P., Pandey, O., Upadhyay, J.: Block-wise non-malleable codes. In: LIPIcs-Leibniz International Proceedings in Informatics, vol. 55. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2016)
12. Chor, B., Goldreich, O.: Unbiased bits from sources of weak randomness and probabilistic communication complexity. In: Proceedings of 26th FOCS, pp. 429–442. IEEE (1985)
13. Chor, B., Goldreich, O.: Unbiased bits from sources of weak randomness and probabilistic communication complexity. SIAM J. Comput. **17**(2), 230–261 (1988)

14. Cleve, R.: Limits on the security of coin flips when half the processors are faulty. In: Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing, pp. 364–369. ACM (1986)
15. Cleve, R., Impagliazzo, R.: Martingales, collective coin flipping and discrete control processes. In other words **1**, 5 (1993)
16. Corrigan-Gibbs, H., Jana, S.: Recommendations for randomness in the operating system, or how to keep evil children out of your pool and other random facts. In: HotOS (2015)
17. Dachman-Soled, D., Lindell, Y., Mahmoody, M., Malkin, T.: On the black-box complexity of optimally-fair coin tossing. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 450–467. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19571-6_27
18. Damgård, I., Faust, S., Mukherjee, P., Venturi, D.: Tamper resilient cryptography without self-destruct. Cryptology ePrint Archive, Report 2013/124 (2013). http://eprint.iacr.org/2013/124
19. Dodis, Y., Ong, S.J., Prabhakaran, M., Sahai, A.: On the (im)possibility of cryptography with imperfect randomness. In: FOCS: IEEE Symposium on Foundations of Computer Science (FOCS) (2004)
20. Dodis, Y.: New imperfect random source with applications to coin-flipping. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) ICALP 2001. LNCS, vol. 2076, pp. 297–309. Springer, Heidelberg (2001). doi:10.1007/3-540-48224-5_25
21. Dodis, Y., Pointcheval, D., Ruhault, S., Vergniaud, D., Wichs, D.: Security analysis of pseudo-random number generators with input:/dev/random is not robust. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, pp. 647–658. ACM (2013)
22. Dodis, Y., Yao, Y.: Privacy with imperfect randomness. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 463–482. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48000-7_23
23. Dwork, C., Rothblum, G.N., Vadhan, S.: Boosting and differential privacy. In: 2010 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 51–60. IEEE (2010)
24. Dziembowski, S., Faust, S., Standaert, F.-X.: Private circuits III: hardware Trojan-resilience via testing amplification. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) 23rd Conference on Computer and Communications Security, ACM CCS 2016, pp. 142–153. ACM Press, Vienna (2016)
25. Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. In: Yao, A.C.-C. (ed.) ICS, pp. 434–452. Tsinghua University Press (2010)
26. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic tamper-proof (ATP) security: theoretical foundations for security against hardware tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24638-1_15
27. Goldwasser, S., Kalai, Y.T., Park, S.: Adaptively secure coin-flipping, revisited. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) ICALP 2015. LNCS, vol. 9135, pp. 663–674. Springer, Heidelberg (2015). doi:10.1007/978-3-662-47666-6_53
28. Gutterman, Z., Pinkas, B., Reinman, T.: Analysis of the Linux random number generator. In: 2006 IEEE Symposium on Security and Privacy, p. 15. IEEE (2006)
29. Haitner, I., Omri, E.: Coin flipping with constant bias implies one-way functions. SIAM J. Comput. **43**(2), 389–409 (2014)
30. Heninger, N., Durumeric, Z., Wustrow, E., Halderman, J.A.: Mining your Ps and Qs: detection of widespread weak keys in network devices. In: USENIX Security Symposium, vol. 8 (2012)

31. Kearns, M., Li, M.: Learning in the presence of malicious errors. SIAM J. Comput. **22**(4), 807–837 (1993)
32. Kiayias, A., Tselekounis, Y.: Tamper resilient circuits: the adversary at the gates. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 161–180. Springer, Heidelberg (2013). doi:10.1007/978-3-642-42045-0_9
33. Lichtenstein, D., Linial, N., Saks, M.: Some extremal problems arising from discrete control processes. Combinatorica **9**(3), 269–287 (1989)
34. Maji, H.K., Prabhakaran, M., Sahai, A.: On the computational complexity of coin flipping. In: 2010 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 613–622. IEEE (2010)
35. Papernot, N., McDaniel, P., Sinha, A., Wellman, M.: Towards the science of security and privacy in machine learning. arXiv preprint arXiv:1611.03814 (2016)
36. Reingold, O., Vadhan, S., Wigderson, A.: A note on extracting randomness from santha-vazirani sources. Unpublished manuscript (2004)
37. Rubinstein, B.I.P., Nelson, B., Huang, L., Joseph, A.D., Lau, S.-H., Rao, S., Taft, N., Tygar, J.D.: Antidote: understanding and defending against poisoning of anomaly detectors. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference, pp. 1–14. ACM (2009)
38. Rubinstein, B.I.P., Nelson, B., Huang, L., Joseph, A.D., Lau, S.-H., Rao, S., Taft, N., Tygar, J.D.: Stealthy poisoning attacks on PCA-based anomaly detectors. ACM SIGMETRICS Perform. Eval. Rev. **37**(2), 73–74 (2009)
39. Santha, M., Vazirani, U.V.: Generating quasi-random sequences from semi-random sources. J. Comput. Syst. Sci. **33**(1), 75–87 (1986)
40. Shen, S., Tople, S., Saxena, P.: 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)
41. Valiant, L.G.: A theory of the learnable. Commun. ACM **27**(11), 1134–1142 (1984)
42. Valiant, L.G.: Learning disjunction of conjunctions. In: IJCAI, pp. 560–566 (1985)
43. Von Neumann, J.: 13. various techniques used in connection with random digits. Appl. Math Ser **12**, 36–38 (1951)
44. Xiao, H., Biggio, B., Brown, G., Fumera, G., Eckert, C., Roli, F.: Is feature selection secure against training data poisoning? In: ICML, pp. 1689–1698 (2015)