Proceedings of Machine Learning Research vol 117:1–22, 2020 31st International Conference on Algorithmic Learning Theory

Adversarially Robust Learning Could Leverage Computational Hardness

Sanjam Garg SANJAMG@BERKELEY.EDU
UC Berkeley

Somesh Jha JHA@CS.WISC.EDU
University of Wisconsin Madison

Saeed Mahloujifar SAEED@VIRGINIA.EDU
University of Virginia

Mohammad Mahmoody MOHAMMAD@VIRGINIA.EDU
University of Virginia

Editors: Aryeh Kontorovich and Gergely Neu

Abstract

Over recent years, devising classification algorithms that are robust to adversarial perturbations has emerged as a challenging problem. In particular, deep neural nets (DNNs) seem to be susceptible to small imperceptible changes over test instances. However, the line of work in *provable* robustness, so far, has been focused on *information theoretic* robustness, ruling out even the *existence* of any adversarial examples. In this work, we study whether there is a hope to benefit from *algorithmic* nature of an attacker that searches for adversarial examples, and ask whether there is *any* learning task for which it is possible to design classifiers that are only robust against *polynomial-time* adversaries. Indeed, numerous cryptographic tasks (e.g. encryption of long messages) can only be secure against computationally bounded adversaries, and are indeed *impossible* for computationally unbounded attackers. Thus, it is natural to ask if the same strategy could help robust learning.

We show that computational limitation of attackers can indeed be useful in robust learning by demonstrating the possibility of a classifier for some learning task for which computational and information theoretic adversaries of bounded perturbations have very different power. Namely, while computationally unbounded adversaries can attack successfully and find adversarial examples with small perturbation, polynomial time adversaries are unable to do so unless they can break standard cryptographic hardness assumptions. Our results, therefore, indicate that perhaps a similar approach to cryptography (relying on computational hardness) holds promise for achieving computationally robust machine learning. On the reverse directions, we also show that the existence of such learning task in which computational robustness beats information theoretic robustness requires computational hardness by implying (average-case) hardness of $\mathbf{NP}$.

1. Introduction

Designing classifiers that are robust to small perturbations to test instances has emerged as a challenging task in machine learning. The goal of robust learning is to design classifiers h that still correctly predicts the true label even if the input x is perturbed minimally to a "close" instance x'. In fact, it was shown (Szegedy et al., 2014; Biggio et al., 2013; Goodfellow et al., 2015) that many learning algorithms, and in particular DNNs, are highly vulnerable to such small perturbations and thus adversarial examples can be successfully found. Since then, the machine learning community

has been actively engaged to address this problem with many new defenses (Papernot et al., 2016; Madry et al., 2018; Biggio and Roli, 2018) and novel and powerful attacks (Carlini and Wagner, 2017; Athalye et al., 2018).

Do adversarial examples always *exist*? This state of affairs suggest that perhaps the existence of adversarial example is due to fundamental reasons that might be inevitable. A sequence of work (Gilmer et al., 2018; Fawzi et al., 2018; Diochnos et al., 2018; Mahloujifar et al., 2019; Shafahi et al., 2018; Dohmatob, 2018) show that for natural theoretical distributions (e.g., isotropic Gaussian of dimension n) and natural metrics over them (e.g., ℓ_0, ℓ_1 or ℓ_2), adversarial examples are inevitable. Namely, the concentration of measure phenomenon (Ledoux, 2001; Milman and Schechtman, 1986) in such metric probability spaces imply that small perturbations are enough to map almost all the instances x into a close x' that is misclassified. This line of work, however, does not yet say anything about "natural" distributions of interest such as images or voice, as the precise nature of such distributions are yet to be understood.

Can lessons from cryptography help? Given the pessimistic state of affairs, researchers have asked if we could use lessons from cryptography to make progress on this problem (Madry, 2018; Goldwasser, 2018; Mahloujifar and Mahmoody, 2018). Indeed, numerous cryptographic tasks (e.g. encryption of long messages) can only be realized against attackers that are computationally bounded. In particular, we know that all encryption methods that use a short key to encrypt much longer messages are insecure against computationally unbounded adversaries. However, when restricted to computationally bounded adversaries this task becomes feasible and suffices for numerous settings. This insight has been extremely influential in cryptography. Nonetheless, despite attempts to build on this insight in the learning setting, we have virtually no evidence on whether this approach is promising. Thus, in this work we study the following question:

> *Could we hope to leverage computational hardness for the benefit of adversarially robust learning by rendering successful attacks computationally infeasible?*

Taking a step in realizing this vision, we provide formal definitions for *computational* variants of robust learning. Following the cryptographic literature, we provide a game based definition of computationally robust learning. Very roughly, a game-based definition consists of two entities: a *challenger* and an *attacker*, that interact with each other. In our case, as the first step the challenger generates independent samples from the distribution at hand, use those samples to train a learning algorithm, and obtain a hypothesis h. Additionally, the challenger samples a fresh challenge sample x from the underlying distribution. Next, the challenger provides the attacker with oracle access to $h(\cdot)$ and x. At the end of this game, the attacker outputs a value x' to the challenger. The attacker declares this execution as a "win" if x' is obtained as a small perturbation of x and leads to a misclassification. We say that the learning is computationally robust as long as no attacker from a class of adversaries can "win" the above game with a probability much better than some base value. (See Definition 1.) This definition is very general and it implies various notions of security by restricting to various classes of attackers. While we focus on polynomially bounded attackers in this paper, we remark that one may also naturally consider other natural classes of attackers based on the setting of interest (e.g. an attacker that can only modify certain part of the image).

What if adversarial examples are actually easy to find? Mahloujifar and Mahmoody (2019) studied this question, and showed that as long as the input instances come from a product distribution, and if the distances are measured in Hamming distance, adversarial examples with sublinear

perturbations can be found in polynomial time. This result, however, did not say anything about other distributions or metrics such as ℓ_∞. Thus, it was left open whether computational hardness could be leveraged in any learning problem to guarantee its robustness.

1.1. Our Results

From computational hardness to computational robustness. In this work, we show that computational hardness can indeed be leveraged to help robustness. In particular, we present a learning problem P that has a classifier h_{P} that is *only* computationally robust. In fact, let Q be *any* learning problem that has a classifier with "small" risk α, but that adversarial examples *exist* for classifier h_{Q} with higher probability $\beta \gg \alpha$ under the ℓ_0 norm (e.g., Q could be *any* of the well-studied problems in the literature with a vulnerable classifier $h_{\mathcal{Q}}$ under norm ℓ_0). Then, we show that there is a "related" problem P and a related classifier h_{P} that has *computational* risk (i.e., risk in the presence of *computationally bounded* tampering adversaries) at most α, but the risk of h_{P} will go up all the way to $\approx \beta$ if the tampering attackers are allowed to be computationally unbounded. Namely, computationally bounded adversaries have a much smaller chance of finding adversarial examples of small perturbations for h_{P} than computationally unbounded attackers do. (See Theorem 6.)

The computational robustness of the above construction relies on allowing the hypothesis to sometimes "detect" tampering and output a special symbol $\star$. The goal of the attacker is to make the hypothesis output a wrong label and *not* get detected. Therefore, we have proved, along the way, that allowing tamper detection can also be useful for robustness. Allowing tamper detection, however, is not always an option. For example a real-time decision making classifier (e.g., classifying a traffic sign) that *has to* output a label, even if it detects that something might be suspicious about the input image. We prove that even in this case, there is a learning problem P with binary labels and a classifier h for P such that computational risk of h is almost zero, while its information theoretic risk is $\approx 1/2$, which makes classifiers' decisions under attack meaningless. (See Theorem 8).

Extension: existence of learning *problems* that are computationally robust. Our result above applies to certain classifiers that "separate" the power of computationally bounded vs. that of computationally unbounded attackers. Doing so, however, does not rule out the possibility of finding information theoretically robust classifiers for the *same* problem. So, a natural question is: can we extend our result to show the existence of learning tasks for which *any* classifier is vulnerable to unbounded attackers, while computationally robust classifiers for that task exist? At first, it might look like an impossible task, in "natural" settings, in which the ground truth function c itself is robust under the allowed amount of perturbations. (For example, in case of image classification, Human is the robust ground truth). Therefore, we cannot simply extend our result in this setting to rule out the existence of robust classifiers, since they might simply exist (unless one puts a limits on the complexity of the learned model, to exclude the ground truth function as a possible hypothesis).

However, one can still formulate the question above in a meaningful way as follows: Can we have a learning task for which any *polynomial time* learning algorithm (with polynomial sample complexity) is forced to produce (with high probability) hypotheses with low robustness against unbounded attacks? Indeed, in this work we also answer this question affirmatively, as a corollary to our main result, by also relying on recent results proved in recent exciting works of (Bubeck et al., 2018c,a; Degwekar and Vaikuntanathan, 2019).

In summary, our work provides credence that perhaps restricting attacks to computationally bounded adversaries holds promise for achieving computationally robust machine learning that relies on computational hardness assumptions as is currently done in cryptography.

From computational robustness back to computational hardness. Our first result shows that computational hardness can be leveraged in some cases to obtain nontrivial computational robustness that beats information theoretic robustness. But how about the reverse direction; are computational hardness assumptions necessary for this goal? We also prove such reverse direction and show that nontrivial computational robustness implies computationally hard problems in $\mathbf{NP}$. In particular, we show that a non-negligible gap between the success probability of computationally bounded vs. that of unbounded adversaries in attacking the robustness of classifiers implies strong average-case hard distributions for class $\mathbf{NP}$. Namely, we prove that if the distribution D of the instances in learning task is efficiently samplable, and if a classifier h for this problem has computational robustness α, information theoretic robustness β, and $\alpha < \beta$, then one can efficiently sample from a distribution S that generates Boolean formulas $\phi \leftarrow S$ that are satisfiable with overwhelming probability, yet no efficient algorithm can find the satisfying assignments of $\phi \leftarrow S$ with a non-negligible probability. (See Theorem 10 for the formal statement.)

What world do we live in? As explained above, our main question is whether adversarial examples could be prevented by relying on computational limitations of the adversary. In fact, even if adversarial examples exist for a classifier, we might be living in either of two completely different worlds. One is a world in which computationally unbounded adversaries can find adversarial examples (almost) whenever they exist and they would be as powerful as information-theoretic adversaries. Another world is one in which machine learning could leverage computational hardness. Our work suggests that computational hardness can potentially help robustness for certain learning problems; thus, we are living in the better world. Whether or not we can achieve *computational* robustness for *practical* problems (such as image classification) that beats their information-theoretic robustness remains an intriguing open question. A related line of work (Bubeck et al., 2018c,a; Degwekar and Vaikuntanathan, 2019) studied other "worlds" that we might be living in, and studied whether adversarial examples are due to the computational hardness of *learning* robust classifiers. They designed learning problems demonstrating that in some worlds, robust classifiers might exist, while they are hard to be obtained efficiently. We note however, that the goal of those works and our work are quite different. They deal with how computational constraints might be an issue and *prevent* the learner from reaching its goal, while our focus is on how such constraints on adversaries can *help* us achieve robustness guarantees that are not achievable information theoretically.

What does our result say about robustifying other natural learning tasks? Our results only show the *existence* of a learning task for which computational robustness is very meaningful. So, one might argue that this is an ad hoc phenomenon that might not have an impact on other practical problems (such as image classification). However, we emphasize that prior to our work, there was *no* provable evidence that computational hardness can play any positive role in robust learning. Indeed, our results also shed light on how computational robustness can potentially be applied to other, perhaps more natural learning tasks. The reason is that the space of all possible ways to tamper a high dimensional vector is *exponentially* large. Lessons from cryptography, and the construction of our learning task proving our main result, suggest that, in such cases, there is potentially a huge gap between the power of computationally bounded vs. unbounded search algorithms. On the other

hand, there are methods proposed by researchers that *seem* to resist attacks that try to find adversarial examples (Madry et al., 2018), while the certified robustness literature is all focused on modeling the adversary as a computationally unbounded entity who can find adversarial examples within a certain distance, so long as they exist (Raghunathan et al., 2018; Wong and Kolter, 2018; Sinha et al., 2018; Wong et al., 2018). Our result shows that, perhaps we shall start to consider *computational* variants of certification methods that focus on computationally bounded adversaries, as by doing so we might be able to prove better robustness bounds for methods that are designed already.

Other related work. In another line of work (Raghunathan et al., 2018; Wong and Kolter, 2018; Sinha et al., 2018; Wong et al., 2018) the notion of *certifiable* robustness was developed to prove robustness for individual test instances. More formally, they aim at providing robustness certificates with bounds ε_x along with a decision $h(x)$ made on a test instance x, with the guarantee that any x' at distance at most ε_x from x is correctly classified. However, these guarantees, so far, are not strong enough to rule out attacks completely, as larger magnitudes of perturbation (than the levels certified) still can fool the classifiers while the instances look the same to the human.

1.1.1. TECHNIQUES

We prove our main result about the possibility of computationally robust classifiers (Theorem 6) by "wrapping" an arbitrary learning problem Q with a vulnerable classifier by adding *computational* certification based on cryptographic digital signatures to test instances. A digital signature scheme (see Definition 12) operates based on two generated keys $(\mathsf{vk}, \mathsf{sk})$, where sk is private and is used for *signing* messages, and vk is public and is used for *verifying* signatures. Such schemes come with the guarantee that a computationally bounded adversary with the knowledge of vk cannot sign new messages on its own, even if it is given signatures on some previous messages. Digital signature schemes can be constructed based on the assumption that one-way functions exist.[1] Below we describe the ideas behind this result in two steps.

Initial Attempt. Suppose D_{Q} is the distribution over $\mathcal{X} \times \mathcal{Y}$ of a learning problem Q with input space $\mathcal{X}$ and label space $\mathcal{Y}$. Suppose D_{Q} had a hypothesis h_{Q} that can predict correct labels reasonably well, $\Pr_{(x,y)\leftarrow D_{\mathsf{Q}}}[h(x) \neq y] \leq \alpha$. Suppose, at the same time, that a (perhaps computationally unbounded) adversary A can perturb test instances like x into a close adversarial example x' that is now likely to be misclassified by h_{Q},

$$\Pr_{(x,y)\leftarrow D_{\mathsf{Q}}}[h(x') \neq y; x' = \mathsf{A}(x)] \geq \beta \gg \alpha.$$

Now we describe a related problem P, its distribution of examples D_{P}, and a classifier h_{P} for P. To sample an example from D_{P}, we first sample $(x, y) \leftarrow D_{\mathsf{Q}}$ and then modify x to $\overline{x} = (x, \sigma_x)$ by attaching a *short* signature $\sigma_x = \mathrm{Sign}(\mathsf{sk}, x)$ to x. The label y of $\overline{x}$ remains the same as that of x. Note that sk will be kept secret to the sampling algorithm of D_{P}. The new classifier h_{P} will rely on the public parameter vk that is available to it. Given an input $\overline{x} = (x, \sigma_x)$, h_{P} first checks its integrity by verifying that the given signature σ_x is valid for x. If the signature verification does not pass, h_{P} rejects the input as adversarial without outputting a label, but if this test passes, h_{P} outputs $h_{\mathsf{Q}}(x)$.

1. Here, we need signature schemes with "short" signatures of poly-logarithmic length over the security parameter. They could be constructed based on exponentially hard one-way functions (Rompel, 1990) by picking the security parameter sub-exponentially smaller that usual and using universal one-way hash functions to hash the message to poly-logarithmic length..

To successfully find an adversarial example $\overline{x}'$ for h_{P} through a small perturbation of $\overline{x} = (x, \sigma)$ sampled as $(\overline{x}, y) \leftarrow D_{\mathsf{P}}$, an adversary A can pursue either of the following strategies. **(I)** One strategy is that A tries to find a new signature $\sigma' \neq \sigma_x$ for the same x, which will constitute as a sufficiently small perturbation as the signature is short. Doing so, however, is not considered a successful attack, as the label of $\overline{x}'$ remains the same as that of the true label of the untampered point $\overline{x}$. **(II)** Another strategy is to perturb the x part of $\overline{x}$ into a close instance x' and then trying to find a correct signature σ' for it, and outputting $\overline{x}' = (x', \sigma')$. Doing so would be a successful attack, because the signature is short, and thus $\overline{x}'$ would indeed be a close instance to $\overline{x}$. However, doing this is computationally infeasible, due to the very security definition of the signature scheme. Note that $(x'\sigma')$ is a forgery for the signature scheme, which a computationally bounded adversary cannot construct because of the security of the underlying signature scheme. This means that the *computational* risk of h_{P} would remain at most α.

We now observe that information theoretic (i.e., computationally unbounded) attackers can succeed in finding adversarial examples for h_{P} with probability at least $\beta \gg \alpha$. In particular, such attacks can first find an adversarial example x' for x (which is possible with probability β over the sampled x), construct a signature σ' for x', and then output (x', σ'). Recall that an unbounded adversary can construct a signature σ' for x' using exhaustive search.

Actual construction. One main issue with the above construction is that it needs to make vk publicly available, as a public parameter to the hypothesis (after it is sampled as part of the description of the distribution D_{P}). Note that it is computationally hard to construct the hypothesis described above without knowing vk. The problem with revealing vk to the learner is that the distribution of examples should come with some extra information other than samples. However, in the classical definition of a learning problem, the learner only has access to samples from the distribution. In fact, if we were allowed to pass some extra information to the learner, we could pass the description of a robust classifier (e.g. the ground truth) and the learning task becomes trivial. The other issue is that the distribution D_{P} is not *publicly samplable* in polynomial time, because to get a sample from D_{P} one needs to use the signing key sk, but that key is kept secret. We resolve these two issues with two more ideas. The first idea is that, instead of generating *one* pair of keys $(\mathsf{vk}, \mathsf{sk})$ for D_{P} and keeping sk_D secret, we can generate a fresh pair of keys $(\mathsf{vk}_x, \mathsf{sk}_x)$ every time that we sample $(x, y) \leftarrow D_{\mathsf{Q}}$ and attach vk_x also to the actual instance $\overline{x} = (x, \sigma_x, \mathsf{vk}_k)$. The modified hypothesis h_{P} also uses this key and verifies (x, σ_x) using vk_x. This way, the distribution D_{P} is publicly samplable, and moreover, there is no need for making vk available as a public parameter. However, this change of the distribution D_{P} introduces a new possible way to attack the scheme and to find adversarial examples. In particular, now the adversary can try to perturb vk_x into a close string vk' for which it knows a corresponding signing key sk', and then use sk' to sign an adversarial example x' for x and output $(x', \sigma', \mathsf{vk}')$. However, to make this attack impossible for the attacker under small perturbations of instances, we use error correction codes and employ an encoding $[\mathsf{vk}_x]$ of the verification key (instead of vk_x) that needs too much change before one can fool a decoder to decode to any other $\mathsf{vk}' \neq \mathsf{vk}_x$. But as long as the adversary cannot change vk_x, the adversary cannot attack the robustness computationally. (See Construction 5.)

To analyze the construction above (see Theorem 6), we note that the computationally bounded adversary would need to change $\Omega(|x|)$ number of bits in $(x, \sigma, [\mathsf{vk}])$ to get $(x', \sigma', [\mathsf{vk}'])$ where $x \neq x'$. This is because the encoded $[\mathsf{vk}]$ would need $\Omega(|x|)$ number of perturbations to change the encoded vk, and if vk remains the same it is hard computationally to find a valid signature. On the

other hand, a computationally unbounded adversary can focus on perturbing x into x' and then forge a short signatures for it, which could be as small as $\mathrm{poly}(\log(|x|))$ perturbations.

Extension to problems, rather than specific classifiers for them. Note that the construction above could be wrapped around any learning problem. In particular, we can pick an original problem that is not (information theoretically) robustly learnable in polynomial time. These problems, which we call them robust-hard are studied recently in (Bubeck et al., 2018c) and (Degwekar and Vaikuntanathan, 2019) where they construct such robust-hard problems to show the effect of computational limitation in robust learning (See Definition 14 and 15) . Here, using their construction as the original learning problem, and wrapping it with our construction, we can strengthen our result and construct a learning problem that is not robustly learnable by *any* polynomial time learning algorithm, yet it has a classifier that is computationally robust. See Corollary 7 for more details.

Computational robustness without tamper detection. The computational robustness of the constructed classifier relies on sometimes detecting tampering attacks and not outputting a label. We give an alternative construction for a setting that the classifier *always* has to output a label. We again use digital signatures and error correction codes as the main ingredient of our construction but in a different way. The main difference is that we have to repeat the signature multiple times to prevent the adversary from changing all of the signatures. The caveat of this construction is that it is no longer a wrapper around an arbitrary learning problem. See Construction 19 for more details.

2. Defining Computational Risk and Computationally Robust Learning

Notation. We use calligraphic letters (e.g., $\mathcal{X}$) for sets and capital non-calligraphic letters (e.g., D) for distributions. By $d \leftarrow D$ we denote sampling d from D. For a randomized algorithm $R(\cdot)$, $y \leftarrow R(x)$ denotes the randomized execution of R on input x outputting y. A classification problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{H})$ is specified by the following components: set $\mathcal{X}$ is the set of possible *instances*, $\mathcal{Y}$ is the set of possible *labels*, $D \in \mathcal{D}$ is a *joint* distribution over $\mathcal{X} \times \mathcal{Y}$, and $\mathcal{H}$ is the space of hypothesis. For simplicity we work with problems that have a single distribution D (e.g., D is the distribution of labeled images from a data set like MNIST or CIFAR-10). A learner L for problem P is an algorithm that takes a dataset $\mathcal{S} \leftarrow D^m$ as input and outputs a hypothesis $h \in \mathcal{H}$. We did not state the loss function explicitly, as we work with classification problems and use the zero-one loss by default. For a learning problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{H})$, the *risk* or *error* of a hypothesis $h \in \mathcal{H}$ is $\mathsf{Risk}_{\mathsf{P}}(h) = \Pr_{(x,y)\leftarrow D}[h(x) \neq y]$. We are usually interested in learning problems $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{H})$ with a specific metric $\mathbf{d}$ defined over $\mathcal{X}$ for the purpose of defining adversarial perturbations of bounded magnitude controlled by $\mathbf{d}$. In that case, we might simply write $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{H})$, but $\mathbf{d}$ is implicitly defined over $\mathcal{X}$. Finally, for a metric $\mathbf{d}$ over $\mathcal{X}$, we let $\mathbf{d}_b(x) = \{x' \mid \mathbf{d}(x, x') \leq b\}$ be the ball of radius b centered at x under the metric $\mathbf{d}$. By default, we work with Hamming distance $\mathsf{HD}(x, x') = |\{i \colon x_i \neq x'_i\}|$, but our definitions can be adapted to any other metrics. We usually work with *families* of problems P_n where n determines the length of $x \in \mathcal{X}_n$ (and thus input lengths of $h \in \mathcal{H}_n, c \in \mathcal{C}_n, \mathbf{d}_n$). We sometimes use a special notation $\Pr[x \leftarrow X; E(x)]$ to define $\Pr_{x\leftarrow X}[E(x)]$ that is the probability of and event E over a random variable X. We also might use a combination of multiple random variables, for examples $\Pr[x \leftarrow X; y \leftarrow Y; E(x, y)]$ denotes the same thing as $\Pr_{x\leftarrow X, y\leftarrow Y}[E(x, y)]$. Order of sampling of X and Y matters Y might depend on X.

Allowing tamper detection. In this work, we expand the standard notion of hypotheses and allow $h \in \mathcal{H}$ to output a special symbol $\star$ as well (without adding $\star$ to $\mathcal{Y}$), namely we have $h \colon \mathcal{X} \mapsto \mathcal{Y} \cup \{\star\}$.

This symbol can be used by the classifier h to denote "out of distribution" points, or any form of tampering, without outputting an actual label. In natural scenarios, $h(x) \neq \star$ when x is not an adversarially tampered instance. However, we allow this symbol to be output by h even in no-attack settings as long as its probability is small enough.

We follow the tradition of game-based security definitions in cryptography (Naor, 2003; Shoup, 2004; Goldwasser and Kalai, 2016; Rogaway and Zhang, 2018). Games are the most common way that security is defined in cryptography. These games are defined between a challenger Chal and an adversary A. Consider the case of a signature scheme. In this case the challenger Chal is a signature scheme Π and an adversary A is given oracle access to the signing functionality (i.e. adversary can give a message m_i to the oracle and obtains the corresponding signature σ_i). Adversary A wins the game if he can provide a valid signature on a message that was not queried to the oracle. The security of the signature scheme is then defined informally as follows: any probabilistic polynomial time/size adversary A can win the game by probability that is bounded by a negligible $n^{-\omega(1)}$ function on the security parameter. We describe a security game for tampering adversaries with bounded tampering budget in HD, but the definition is more general and can be used for other adversary classes.

Definition 1 (Security game of adversarially robust learning) *Let $\mathsf{P}_n = (\mathcal{X}_n, \mathcal{Y}_n, D_n, \mathcal{H}_n)$ be a classification problem where the components are parameterized by n. Let L be a learning algorithm with sample complexity $m = m(n)$ for P_n. Consider the following game between a challenger Chal, and an adversary A with tampering budget $b = b(n)$.*

1. Chal *samples m i.i.d. examples $\mathcal{S} \leftarrow D_n^m$ and gets hypothesis $h \leftarrow L(\mathcal{S})$ where $h \in \mathcal{H}_n$.*
2. Chal *then samples a test example $(x, y) \leftarrow D_n$ and sends (x, y) to the adversary A.*
3. *Having oracle access (or oracle gates, in case of circuits) to hypothesis h and a sampler for D_n, the adversary obtains the adversarial instance $x' \leftarrow \mathsf{A}^{h(\cdot), D_n}(x)$ and outputs x'.*

Winning conditions: *In case $x = x'$, the adversary A wins if $h(x) \neq y$,*[2] *and in case $x \neq x'$, the adversary wins if all the following hold: (1) $\mathsf{HD}(x, x') \leq b$, (2) $h(x') \neq y$, and (3) $h(x') \neq \star$.*

Why separating winning conditions for $x = x'$ from $x \neq x'$? One might wonder why we separate the winning condition for the two cases of $x = x'$ and $x \neq x'$. The reason is that $\star$ is supposed to capture tamper detection. So, if the adversary does not change x and the hypothesis outputs $h(x) = \star$, this is an error, and thus should contribute to the risk. More formally, when we evaluate risk, we have $\mathsf{Risk}_{\mathsf{P}}(h) = \Pr_{(x,y)\leftarrow D}[h(x) \neq y]$, which implicitly means that outputting $\star$ contributes to the risk. However, if adversary's perturbs to $x' \neq x$ leads to $h(x') = \star$, it means the adversary has *not* succeeded in its attack, because the tampering is detected. In fact, if we simply require the other 3 conditions to let adversary win, the notion of "adversarial risk" (see Definition 2) might be even less than the normal risk, which is counter intuitive.

Alternative definitions of winning for the adversary. The winning condition for the adversary could be defined in other ways as well. In our Definition 1, the adversary wins if $\mathbf{d}(x, x') \leq b$ and $h(x') \neq y$. This condition is inspired by the notion of corrupted input (Feige et al., 2015), is extended to metric spaces in (Madry et al., 2018), and is used in and many subsequent works. An alternative definition for adversary's goal, formalized in (Diochnos et al., 2018) and used in (Gilmer et al., 2018;

2. Note that, if $h(x) \neq y$, without loss of generality, the adversary A can output $x' = x$

Diochnos et al., 2018; Bubeck et al., 2018a; Degwekar and Vaikuntanathan, 2019) requires $h(x')$ to be different from the true label of x' (rather than x). This condition requires the misclassification of x', and thus, x' would belong to the "error-region" of h. Namely, if we let $c(x) = y$ be the ground truth function, the error-region security game requires $h(x') \neq c(x')$. Another stronger definition of adversarial risk is given by Suggala et al. (2018) in which the requirement condition requires both conditions: (1) the ground truth should not change $c(x) = c(x')$, and that (2) x' is misclassified. For natural distributions like images or voice, where the ground truth is robust to small perturbations, all these three definitions for adversary's winning are equivalent.

Stronger attack models. In the attack model of Definition 1, we only provided the label y of x to the adversary and also give her the sample oracle from D_n. A stronger attacker can have access to the "concept" function $c(x)$ which is sampled from the distribution of y given x (according to D_n). This concept oracle might not be efficiently computable, even in scenarios that D_n is efficiently samplable. In fact, even if D_n is not efficiently samplable, just having access to a large enough pool of i.i.d. sampled data from D_n is enough to run the experiment of Definition 1. In alternative winning conditions (e.g., the error-region definition) for Definition 1 discussed above, it makes more sense to also include the ground truth concept oracle $c(\cdot)$ given as oracle to the adversary, as the adversary needs to achieve $h(x') \neq c(x')$. Another way to strengthen the power of adversary is to give him non-black-box access to the components of the game (see Papernot et al. (2017)). In definition 1, by default, we model adversaries who have black-box access to $h(\cdot), D_n$, but one can define non-black-box (white-box) access to each of $h(\cdot), D_n$, if they are polynomial size objects.

Diochnos et al. (2018) focused on bounded perturbation adversaries that are unbounded in their running time and formalized notions of of adversarial risk for a given hypothesis h with respect to the b-perturbing adversaries. Using Definition 1, in Definition 2, we retrieve the notions of standard risk, adversarial risk, and its (new) computational variant.

Definition 2 (Adversarial risk of hypotheses and learners) *Suppose L is a learner for a problem $\mathsf{P} = (\mathcal{X}, \mathcal{Y}, D, \mathcal{H})$. For a class of attackers $\mathcal{A}$ we define*

$$\mathsf{AdvRisk}_{\mathsf{P},\mathcal{A}}(L) = \sup_{\mathsf{A}\in\mathcal{A}} \Pr[\mathsf{A} \textit{ wins}]$$

where the winning is in the experiment of Definition 1. When the attacker A is fixed, we simply write $\mathsf{AdvRisk}_{\mathsf{P},\mathsf{A}}(L) = \mathsf{AdvRisk}_{\mathsf{P},\{\mathsf{A}\}}(L)$. For a trivial attacker I who outputs $x' = x$, it holds that $\mathsf{Risk}_{\mathsf{P}}(L) = \mathsf{AdvRisk}_{\mathsf{P},I}(L)$. When $\mathcal{A}$ includes attacker that are only *bounded by b perturbations, we use notation $\mathsf{AdvRisk}_{\mathsf{P},b}(L) = \mathsf{AdvRisk}_{\mathsf{P},\mathcal{A}}(L)$, and when the adversary is further restricted to all s-size (oracle-aided) circuits, we use notation $\mathsf{AdvRisk}_{\mathsf{P},b,s}(L) = \mathsf{AdvRisk}_{\mathsf{P},\mathcal{A}}(L)$. When L is a learner that outputs a fixed hypothesis h, by substituting h with L, we obtain the following similar notions for h, which will be denoted as $\mathsf{Risk}_{\mathsf{P}}(h)$, $\mathsf{AdvRisk}_{\mathsf{P},\mathcal{A}}(h)$, $\mathsf{AdvRisk}_{\mathsf{P},b}(h)$, and $\mathsf{AdvRisk}_{\mathsf{P},b,s}(h)$.*

Definition 3 (Computationally robust learners and hypotheses) *Let $\mathsf{P}_n = (\mathcal{X}_n, \mathcal{Y}_n, D_n, \mathcal{H}_n)$ be a family of classification parameterized by n. We say that a learning algorithm L is a* computationally robust *learner with risk at most $R = R(n)$ against $b = b(n)$-perturbing adversaries, if for any polynomial $s = s(n)$, there is a negligible function $\mathrm{negl}(n) = n^{-\omega(1)}$ such that*

$$\mathsf{AdvRisk}_{\mathsf{P}_n,b,s}(L) \leq R(n) + \mathrm{negl}(n).$$

Note that the size of circuit used by the adversary controls its computational power and that is why we are enforcing it to be a polynomial. Again, when L is a learner that outputs a fixed hypothesis

h_n for each n, we say that the family h_n is a computationally robust hypothesis with risk at most $R = R(n)$ against $b = b(n)$-perturbing adversaries, if L is so. In both cases, we might simply say that L (or h) has computational risk *at most $R(n)$.*

Remark 4 (Alternative definition without the negligible term for concrete adversary runtime) *We remark that, when the class of adversary is a finite set, and when we work with a concrete setting of parameter (as opposed to the asymptotic setting of Definition 2) one can opt to work with concrete bounds and a version that drops the negligible probability* negl *on the right hand side of the inequality and asks for the probability of winning to be simply stated as* $\mathsf{AdvRisk}_{\mathsf{P}_{n,b,s}}(L) \leq R(n)$ *for s-sized oracle-aided circuit adversaries or s-time oracle-aided Turing machines. However, in the asymptotic setting, one can work with very large polynomials for small security parameters, in which case there is little difference between information theoretic adversaries versus computationally bounded ones. In that case, the negligible additive term will subsume any large advantage that such adversaries might have for small security parameters. In this work, we opt to work with the above asymptotic definition together with the negligible additive term. Moreover, the negligible probability usually comes up in computational reductions, and hence it simplifies the statement of our theorems, but we emphasize that both forms of the definition of computational risk (for concert as well as asymptotic settings) are equally appealing and valid on their own.*

PAC learning under computationally bounded tampering adversaries. Recently, several works studied generalization under adversarial perturbations from a theoretical perspective (Bubeck et al., 2018b; Cullina et al., 2018; Feige et al., 2018; Attias et al., 2018; Khim and Loh, 2018; Yin et al., 2018; Montasser et al., 2019; Diochnos et al., 2019), and hence they implicitly or explicitly revisited the "probably approximately corect" (PAC) learning framework of Valiant (2013) under adversarial perturbations. Here we comment that, one can derive variants of those definitions for *computationally bounded* attackers, by limiting their adversaries as done in our Definition 3. In particular, we call a learner L an (ε, δ) PAC learner for a problem P and computationally bounded b-perturbing adversaries, if with probability $1 - \delta$, L outputs a hypothesis h that has computational risk at most ε.

Discussion on falsifiability of computational robustness. If the learner L is polynomial time, and that the distribution D_n is samplable in polynomial time (e.g., by sampling y first and then using a generative model to generate x for y), then the the computational robustness of learners as defined based on Definitions 3 and 1 is a "falsifiable" notion of security as defined by Naor (2003). Namely, if an adversary claims that it can break the computational robustness of the learner L, it can prove so in polynomial time by participating in a challenge-response game and winning in this game with a noticeable (non-negligible) probability more than $R(n)$. This feature is due to the crucial property of the challenger in Definition 1 that is a polynomial time algorithm itself, and thus can be run efficiently. Not all security games have efficient challengers (e.g., see Pandey et al. (2008)).

3. From Computational Hardness to Computational Robustness

In this section, we will first prove our main result that shows the existence of a learning problem with classifiers that are only computationally robust. We first prove our result by starting from *any* hypothesis that is vulnerable to adversarial examples; e.g., this could be any of the numerous algorithms shown to be susceptible to adversarial perturbations. Our constructions use error correction codes and cryptographic signatures. For definitions of these notions refer to section A.

3.1. Computational Robustness with Tamper Detection

Our first construction uses hypothesis with tamper detection (i.e, output $\star$ capability). This construction is based on cryptographic signature schemes with short (polylogarithmic) signatures.

Construction 5 *Let* $\mathsf{Q} = (\{0,1\}^d, \mathcal{Y}, D, \mathcal{H})$ *be a learning problem and* $h \in \mathcal{H}$ *a classifier for* Q. *We construct a family of learning problems* P_n *(based on the fixed problem* Q*) with a family of classifiers* h_n. *In our construction we use signature scheme* $(\mathrm{KGen}, \mathrm{Sign}, \mathsf{Verify})$ *for which the bit-length of* vk *is* λ *and the bit-length of signature is* $\ell(\lambda) = \mathrm{polylog}(\lambda)$ [3]. *We also use an error correction code* $(\mathsf{Encode}, \mathsf{Decode})$ *with code rate* $\mathsf{cr} = \Omega(1)$ *and error rate* $\mathsf{er} = \Omega(1)$.

1. *The space of instances for* P_n *is* $\mathcal{X}_n = \{0,1\}^{n+d+\ell(n)}$.
2. *The set of labels is* $\mathcal{Y}_n = \mathcal{Y}$.
3. *The distribution* D_n *is defined by the following process: first sample* $(x, y) \leftarrow D$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathrm{KGen}(1^{n \cdot \mathsf{cr}})$, $\sigma \leftarrow \mathrm{Sign}(\mathsf{sk}, x)$, *then encode* $[\mathsf{vk}] = \mathsf{Encode}(\mathsf{vk})$ *and output* $((x, \sigma, [\mathsf{vk}]), y)$.
4. *The classifier* $h_n \colon \mathcal{X}_n \to \mathcal{Y}_n$ *is defined as*

$$h_n(x, \sigma, [\mathsf{vk}]) = \begin{cases} h(x) & \text{if } \mathsf{Verify}\left(\mathsf{Decode}([\mathsf{vk}]), x, \sigma\right), \\ \star & \text{otherwise.} \end{cases}$$

Theorem 6 *For family* P_n *of Construction 5, the family of classifiers* h_n *is computationally robust with adversarial risk at most* $\mathsf{Risk}_Q(h) = \alpha$ *against adversaries with budget* $\mathsf{er} \cdot n$. *(Recall that* er *is the error rate of the error correction code.) On the other hand* h_n *is not robust against information theoretic adversaries of budget* $b + \ell(n)$, *if* h *itself is not robust to* b *perturbations:*

$$\mathsf{AdvRisk}_{\mathsf{P}_n, b+\ell(n)}(h_n) \geq \mathsf{AdvRisk}_{\mathsf{Q},b}(h).$$

Theorem 6 means that, the h_n is computationally robust for adversarial budget as large as $\Omega(n)$ (if we choose a code with constant error correction rate) while it has small information theoretic adversarial robustness for budget value as small as $b + \mathrm{polylog}(n) \leq \mathrm{polylog}(n)$ (note that b is a constant here) if we choose a signature scheme with short signatures of poly-logarithmic length.

For proof of the above theorem, see Appendix B. Now we state the following corollary about robust-hard learning problems that are defined in Appendix A.

Corollary 7 *If the underlying problem* Q *in Construction 5 is robust-hard w.r.t sublinear budget* $b(n)$, *then for* any *polynomial learning algorithm* L *for* P_n *we have*

$$\mathsf{AdvRisk}_{\mathsf{P}_n, b+\ell(n)}(L) \geq 1 - \mathrm{negl}(n).$$

On the other hand, the family of classifiers h_n *for* P_n *is computationally robust with risk at most* α *against adversaries with linear budget.*

The above corollary follows from Theorem 6 and definition of robust-hard learning problems. The significance of this corollary is that it provides an example of a learning problem that could not be learnt robustly with *any* polynomial time algorithm. However, the same problem has a classifier that is robust against computationally bounded adversaries. This construction uses a robust-hard learning problem that is proven to exist based on cryptographic assumptions (Bubeck et al., 2018c; Degwekar and Vaikuntanathan, 2019).

3. Such signatures exist assuming exponentially hard one-way functions (Rompel, 1990).

3.2. Computational Robustness without Tamper Detection

The following theorem shows an alternative construction that is incomparable to Construction 5, as it does not use any tamper detection. See Construction 19 in Appendix B for more details.

Theorem 8 *For family P_n of Construction 19, the family of classifiers h_n has risk 0 and is computationally robust with risk at most 0 against adversaries of budget $\mathsf{er} \cdot n$. On the other hand h_n is not robust against information theoretic adversaries of budget $\ell(n)$:*

$$\mathsf{AdvRisk}_{\mathsf{P}_n,\ell(n)}(h_n) \geq 1/2.$$

Note that reaching adversarial risk $1/2$ makes the classifier's decisions meaningless as a random coin toss achieves this level of accuracy.

4. Average-Case Hardness of NP from Computational Robustness

In this section, we show a converse result to those in Section 3, going from useful computational robustness to deriving computational hardness. Namely, we show that if there is a learning problem whose computational risk is noticeably more than its information theoretic risk, then **NP** is hard even on average.

Definition 9 (Hard samplers for NP) *For the following definition, A Boolean formula ϕ over some Boolean variables $x_1, \ldots, x_k$ is satisfiable, if there is an assignment to $x_1, \ldots, x_k \in \{0,1\}$, for which ϕ evaluates to 1 (i.e, TRUE). We use some standard canonical encoding of such Boolean formulas and fix it, and we refer to $|\phi|$, the size of ϕ, as the bit-length of this representation for formula ϕ. Let* SAT *be the language/set of all satisfiable Boolean formulas. Suppose $S(1^n, r)$ is a polynomial time randomized algorithm that takes 1^n and randomness r, runs in time $\mathrm{poly}(n)$, and outputs Boolean formulas of size $\mathrm{poly}(n)$. We call S a* hard (instance) sampler *for* **NP** *if,*

1. *For a negligible function* negl *it holds that $\Pr_{\phi \leftarrow S}[\phi \in \mathrm{SAT}] = 1 - \mathrm{negl}(n)$.*
2. *For every poly-size circuit A, there is a negligible function* negl*, such that*

$$\Pr_{\phi \leftarrow S, t \leftarrow \mathsf{A}(\phi)}[\phi(t) = 1] = \mathrm{negl}(n).$$

The following theorem is stated for computationally robust learning, but the same proof holds for computationally robust hypotheses as well. See Appendix C for proof of the theorem.

Theorem 10 (Hardness of NP from computational robustness) *Let $\mathsf{P}_n = (\mathcal{X}_n, \mathcal{Y}_n, D_n, \mathcal{H}_n)$ be a learning problem. Suppose there is a (uniform) learning algorithm L for P_n such that:*

1. *L is* computationally *robust with risk at most α under $b(n)$-perturbations.*
2. *$\mathsf{AdvRisk}_{\mathsf{P}_n,b(n)}(L) \geq \beta(n)$; i.e., information-theoretic adversarial risk of L is at least $\beta(n)$.*
3. *$\beta(n) - \alpha \geq \varepsilon$ for $\varepsilon = 1/\mathrm{poly}(n)$.*
4. *D_n is efficiently samplable by algorithm S.*
5. *For any $x, x' \in \mathcal{X}_n$ checking $\mathbf{d}(x, x') \leq b(n)$ is possible in polynomial time.*

Then, there is a hard sampler for **NP**.

References

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, 2018.

Idan Attias, Aryeh Kontorovich, and Yishay Mansour. Improved generalization bounds for robust learning. *arXiv preprint arXiv:1810.02180*, 2018.

Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion Attacks against Machine Learning at Test Time. In *ECML/PKDD*, pages 387–402, 2013.

Sébastien Bubeck, Yin Tat Lee, Eric Price, and Ilya Razenshteyn. Adversarial examples from cryptographic pseudo-random generators. *arXiv preprint arXiv:1811.06418*, 2018a.

Sébastien Bubeck, Eric Price, and Ilya Razenshteyn. Adversarial examples from computational constraints. *arXiv preprint arXiv:1805.10204*, 2018b.

Sébastien Bubeck, Eric Price, and Ilya Razenshteyn. Adversarial examples from computational constraints. *arXiv preprint arXiv:1805.10204*, 2018c.

Ran Canetti, Shai Halevi, and Michael Steiner. Hardness amplification of weakly verifiable puzzles. In *Theory of Cryptography Conference*, pages 17–33. Springer, 2005.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pages 39–57. IEEE, 2017.

Daniel Cullina, Arjun Nitin Bhagoji, and Prateek Mittal. Pac-learning in the presence of adversaries. In *Advances in Neural Information Processing Systems*, pages 230–241, 2018.

Akshay Degwekar and Vinod Vaikuntanathan. Computational limitations in robust classification and win-win results. *arXiv preprint arXiv:1902.01086*, 2019.

Dimitrios Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. Adversarial risk and robustness: General definitions and implications for the uniform distribution. In *Advances in Neural Information Processing Systems*, pages 10359–10368, 2018.

Dimitrios I Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. Lower bounds for adversarially robust pac learning. *arXiv preprint arXiv:1906.05815*, 2019.

Elvis Dohmatob. Limitations of adversarial robustness: strong no free lunch theorem. *arXiv preprint arXiv:1810.04065*, 2018.

Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. Adversarial vulnerability for any classifier. *arXiv preprint arXiv:1802.08686*, 2018.

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

Uriel Feige, Yishay Mansour, and Robert E Schapire. Robust inference for multiclass classification. In *Algorithmic Learning Theory*, pages 368–386, 2018.

Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.

Oded Goldreich. *Foundations of cryptography: volume 1, basic tools*. Cambridge university press, 2007.

Shafi Goldwasser. From idea to impact, the crypto story: What's next? *IACR Distinguished Lecture at CRYPTO conference*, August 2018.

Shafi Goldwasser and Yael Tauman Kalai. Cryptographic assumptions: A position paper. In *Theory of Cryptography Conference*, pages 505–522. Springer, 2016.

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

Thomas Holenstein and Grant Schoenebeck. General hardness amplification of predicates and puzzles. In *Theory of Cryptography Conference*, pages 19–36. Springer, 2011.

Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. Chapman and Hall/CRC, 2014.

Justin Khim and Po-Ling Loh. Adversarial risk bounds for binary classification via function transformation. *arXiv preprint arXiv:1810.09519*, 2018.

Michel Ledoux. *The Concentration of Measure Phenomenon*. Number 89 in Mathematical Surveys and Monographs. American Mathematical Society, 2001.

Aleksander Madry. Machine learning and security: The good, the bad, and the hopeful. *Talk given at workshop: Beyond Crypto, A TCS Perspective*, August 2018.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083; to appear in International Conference on Learning Representations (ICLR)*, 2018.

Saeed Mahloujifar and Mohammad Mahmoody. Can adversarially robust learning leverage computational hardness? *arXiv preprint arXiv:1810.01407*, 2018.

Saeed Mahloujifar and Mohammad Mahmoody. Can adversarially robust learning leveragecomputational hardness? In *Algorithmic Learning Theory*, pages 581–609, 2019.

Saeed Mahloujifar, Dimitrios I. Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. *AAAI*, 2019.

Vitali D Milman and Gideon Schechtman. *Asymptotic theory of finite dimensional normed spaces*, volume 1200. Springer Verlag, 1986.

Omar Montasser, Steve Hanneke, and Nathan Srebro. Vc classes are adversarially robustly learnable, but only improperly. *arXiv preprint arXiv:1902.04217*, 2019.

Moni Naor. On cryptographic assumptions and challenges. In *Annual International Cryptology Conference*, pages 96–109. Springer, 2003.

Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In *Annual International Cryptology Conference*, pages 57–74. Springer, 2008.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

Phillip Rogaway and Yusi Zhang. Simplifying game-based definitions. In *Annual International Cryptology Conference*, pages 3–32. Springer, 2018.

John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 387–394. ACM, 1990.

Ali Shafahi, W Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? *arXiv preprint arXiv:1809.02104*, 2018.

Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. https://eprint.iacr.org/2004/332.

Aman Sinha, Hongseok Namkoong, and John Duchi. Certifiable distributional robustness with principled adversarial training. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hk6kPgZA-.

Arun Sai Suggala, Adarsh Prasad, Vaishnavh Nagarajan, and Pradeep Ravikumar. On adversarial risk and training. *arXiv preprint arXiv:1806.02924*, 2018.

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

Leslie Valiant. *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*. Basic Books, Inc., New York, NY, USA, 2013.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5283–5292, 2018.

Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. Scaling provable adversarial defenses. *arXiv preprint arXiv:1805.12514*, 2018.

Andrew C Yao. Theory and application of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, pages 80–91. IEEE, 1982.

Dong Yin, Kannan Ramchandran, and Peter Bartlett. Rademacher complexity for adversarially robust generalization. *arXiv preprint arXiv:1810.11914*, 2018.

Appendix A. Useful Tools

Here, we define the notions of one-way function, one-time signature and error correcting code.

Definition 11 (One-way function) *A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is one-way if it can be computed in polynomial time and the inverse of f is hard to compute. Namely, there is a polynomial time algorithm M such that*

$$\Pr[x \leftarrow \{0,1\}^n; M(x) = f(x)] = 1$$

and for any polynomial time algorithm A there is a negligible function $\mathrm{negl}(\cdot)$ such that we have,

$$\Pr[x \leftarrow \{0,1\}^n; y = f(x); f(\mathsf{A}(y)) = x] \leq \mathrm{negl}(|x|).$$

The assumption that one-way functions exist is standard and omnipresent in cryptography as a *minimal* assumption, as many cryptographic tasks imply the existence of OWFs (Goldreich, 2007; Katz and Lindell, 2014).

Definition 12 (One-time signature schemes) *A one-time signature scheme $S = (\mathrm{KGen}, \mathrm{Sign}, \mathsf{Verify})$ consists of three probabilistic polynomial-time algorithms as follows:*

- $\mathrm{KGen}(1^\lambda)$[4] $\to (\mathsf{sk}, \mathsf{vk})$
- $\mathrm{Sign}(\mathsf{sk}, m) \to \sigma$
- $\mathsf{Verify}(\mathsf{vk}, \sigma, m) \to \{0,1\}$

which satisfy the following properties:

1. **Completeness:** *For every m*

$$\Pr[(sk, vk) \leftarrow \mathrm{KGen}(1^\lambda); \sigma \leftarrow \mathrm{Sign}(sk, m); \mathsf{Verify}(\mathsf{vk}, \sigma, m) = 1] = 1.$$

4. By 1^λ we mean an string of bits of size λ that is equal to 1 at each location. Note that λ is the security parameter that controls the security of the scheme. As λ increases the task of finding a forgery for a signature becomes harder.

2. **Unforgeability:** *For every positive polynomial s, for every λ and every pair of circuits (A_1, A_2) with size $s(\lambda)$ the following probability is negligible in λ:*

$$\begin{aligned}\Pr[&(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathrm{KGen}(1^\lambda);\\ &(m, st) \leftarrow A_1(1^\lambda, \mathsf{vk});\\ &\sigma \leftarrow \mathrm{Sign}(\mathsf{sk}, m);\\ &(m', \sigma') \leftarrow A_2(1^\lambda, \mathsf{vk}, st, m, \sigma);\\ &m \neq m' \wedge \mathsf{Verify}(\mathsf{vk}, \sigma', m') = 1] \leq \mathrm{negl}(\lambda).\end{aligned}$$

Definition 13 (Error correction codes) *An error correction code with code rate α and error rate β consists of two algorithms* Encode *and* Decode *as follows.*

- *The encode algorithm* Encode *takes a Boolean string m and outputs a Boolean string c such that $|c| = |m|/\alpha$.*
- *The decode algorithm* Decode *takes a Boolean string c and outputs either $\perp$ or a Boolean string m. It holds that for all $m \in \{0,1\}^*$, $c = \mathsf{Encode}(m)$ and c' where $\mathsf{HD}(c, c') \leq \beta \cdot |c|$, it holds that $\mathsf{Decode}(c') = m$.*

Bellow we formally define the notion of robust-hard learning problems which captures the inherent vulnerability of a learning problem to adversarial attacks due to computational limitations of the learning algorithm. This definition are implicit in works of (Degwekar and Vaikuntanathan, 2019; Bubeck et al., 2018c). In Section 3, we use these robust-hard problems to construct learning problems that are inherently non-robust in presence of computationally unbounded adversaries but have robust classifiers against computationally bounded adversaries.

Definition 14 (Robust-hard learning problems) *A learning problem $\mathsf{P}_n = (\mathcal{X}_n, \mathcal{Y}_n, D_n, \mathcal{H}_n)$ is robust-hard w.r.t budget $b(n)$ if for any learning algorithm L that runs in $\mathrm{poly}(n)$ we have*

$$\mathsf{AdvRisk}_{\mathsf{P}_n, b(L)} \geq 1 - \mathrm{negl}(n).$$

Theorem 15 (Degwekar and Vaikuntanathan (2019); Bubeck et al. (2018c)) *There exist a Learning problem $\mathsf{P}_n = (\mathcal{X}_n, \mathcal{Y}_n, D_n, \mathcal{H}_n)$ and a sub-linear budget $b(n)$ such that P_n is robust-hard w.r.t b unless one-way functions do not exist.*

Appendix B. Proof of Theorems 6 and 8 and Constrcution 19

Proof (of Theorem 6) We first prove the following claim about the risk of h_n.

Claim 16 *For problem P_n we have*

$$\mathsf{Risk}_{\mathsf{P}_n}(h_n) = \mathsf{Risk}_{\mathsf{Q}}(h) = \alpha.$$

Proof The proof follows from the completeness of the signature scheme. We have,

$$\begin{aligned}\mathsf{Risk}_{\mathsf{P}_n}(h_n) &= \Pr[((x, \sigma, [\mathsf{vk}]), y) \leftarrow D_n;\ h_n(x, \sigma, [\mathsf{vk}]) \neq y]\\ &= \Pr[(x, y) \leftarrow D;\ h(x) \neq y] = \mathsf{Risk}_{\mathsf{Q}}(h).\end{aligned}$$

■

Now we prove the computational robustness of h_n.

Claim 17 *For family* P_n*, and for any polynomial* $s(\cdot)$ *there is a negligible function* negl *such that for all* $n \in \mathbb{N}$

$$\mathsf{AdvRisk}_{\mathsf{P}_n,\mathsf{er}\cdot n,s}(h_n) \leq \alpha + \mathrm{negl}(n).$$

Proof Let $\mathsf{A}_{\{n\in\mathbb{N}\}}$ be the family of circuits maximizing the adversarial risk for h_n for all $n \in \mathbb{N}$. We build a sequence of circuits $\mathsf{A}^1_{\{n\in\mathbb{N}\}}$, $\mathsf{A}^2_{\{n\in\mathbb{N}\}}$ such that A^1_n and A^2_n are of size at most $s(n) + poly(n)$. A^1_n just samples a random $(x, y) \leftarrow D$ and outputs (x, y). A^2_n gets x, σ and vk, calls A_n to get $(x', \sigma', \mathsf{vk}') \leftarrow \mathsf{A}_n((x, \sigma, [\mathsf{vk}]), y)$ and outputs (x', σ'). Note that A^2_n can provide all the oracles needed to run A_n if the sampler from D, h and c are all computable by a circuit of polynomial size. Otherwise, we need to assume that our signature scheme is secure with respect to those oracles and the proof will follow. We have,

$$\begin{aligned}\mathsf{AdvRisk}_{\mathsf{P}_n,\mathsf{er}\cdot n,s}(h_n) = \Pr[&((x, \sigma, [\mathsf{vk}]), y) \leftarrow D_n;\ (x', \sigma', \mathsf{vk}') \leftarrow A((x, \sigma, [\mathsf{vk}]), y));\\ &(x', \sigma', \mathsf{vk}') \in \mathsf{HD}_{\mathsf{er}\cdot n}(x, \sigma, [\mathsf{vk}]) \wedge h_n(x', \sigma', \mathsf{vk}') \neq \star \wedge h_n(x', \sigma', \mathsf{vk}') \neq y].\end{aligned}$$

Note that $(x', \sigma', \mathsf{vk}') \in \mathsf{HD}_{\mathsf{er}\cdot n}(x, \sigma, [\mathsf{vk}])$ implies that $\mathsf{Decode}(\mathsf{vk}') = \mathsf{vk}$ based on the error rate of the error correcting code. Also $h_n(x', \sigma', \mathsf{vk}') \neq \star$ implies that σ' is a valid signature for x' under verification key vk. Therefore, we have,

$$\begin{aligned}&\mathsf{AdvRisk}_{\mathsf{er}\cdot n,s}(h_n)\\ &\leq \Pr[(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathrm{KGen}(1^n);\ (x, y) \leftarrow \mathsf{A}_1(1^n);\ \sigma \leftarrow \mathrm{Sign}(\mathsf{sk}, x);\ (x', \sigma') \leftarrow \mathsf{A}_2(x, \sigma, \mathsf{vk});\\ &\qquad \mathsf{Verify}(\mathsf{vk}, x', \sigma') \wedge h_n(x', \sigma', [\mathsf{vk}]) \neq y]\\ &\leq \Pr[(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathrm{KGen}(1^n);\ x \leftarrow \mathsf{A}_1(1^n);\ \sigma \leftarrow \mathrm{Sign}(\mathsf{sk}, x);\ (x', \sigma') \leftarrow \mathsf{A}_2(x, \sigma, \mathsf{vk});\\ &\qquad \mathsf{Verify}(\mathsf{vk}, x', \sigma') \wedge x' \neq x] + \mathsf{Risk}_{\mathsf{P}_n}(h_n).\end{aligned}$$

Thus, by the unforgeability of the one-time signature scheme we have

$$\mathsf{AdvRisk}_{\mathsf{P}_n,\mathsf{er}\cdot n,s}(h_n) \leq \mathsf{Risk}_{\mathsf{P}_n}(h_n) + \mathrm{negl}(n),$$

which by Claim 16 implies

$$\mathsf{AdvRisk}_{\mathsf{er}\cdot n,s}(h_n) \leq \alpha + \mathrm{negl}(n).$$

■

Now we show that h_n is not robust against computationally unbounded attacks.

Claim 18 *For family* P_n *and any* $n, b \in \mathbb{N}$ *we have*

$$\mathsf{AdvRisk}_{\mathsf{P}_n,b+\ell(n)}(h_n) \ \geq\ \mathsf{AdvRisk}_{\mathsf{Q},b(h)}.$$

Proof For any $((x, \sigma, [\mathsf{vk}]), y)$ define $A(x, \sigma, [\mathsf{vk}]) = (x', \sigma', [\mathsf{vk}])$ where x' is the closes point to x where $h(x) \neq y$ and σ' is a valid signature such that $\mathsf{Verify}(\mathsf{vk}, x^*, \sigma') = 1$. Based on the fact that the size of signature is $\ell(n)$, we have $\mathsf{HD}(A(x, \sigma, [\mathsf{vk}]), (x, \sigma, [\mathsf{vk}])) \leq \ell(n) + \mathsf{HD}(x, x')$. Also, it is clear that $h_n(A(x, \sigma, [\mathsf{vk}])) \neq \star$ because σ' is a valid signature. Also, $h_n(A(x, \sigma, [\mathsf{vk}])) \neq$

$c_n(A(x,\sigma,[\mathsf{vk}]))$. Therefore we have

$$\begin{aligned}
&\mathsf{AdvRisk}_{\mathsf{P}_n,b+\ell(n)}(h_n)\\
&= \Pr[((x,\sigma,[\mathsf{vk}]),y)\leftarrow D_n;\\
&\quad \exists(x',\sigma')\in \mathsf{HD}_{b+\ell(n)}(x,\sigma), h(x')\neq y\wedge h(x')\neq\star\wedge \mathsf{Verify}(\mathsf{vk},\sigma',x')]\\
&\geq \Pr[((x,\sigma,[\mathsf{vk}]),y)\leftarrow D_n;\exists x'\in \mathsf{HD}_b(x), h(x')\neq y\wedge h(x')\neq\star]\\
&= \mathsf{AdvRisk}_{\mathsf{Q},b}(h).
\end{aligned}$$

■

This concludes the proof of Theorem 6. ■

Construction 19 (Computational robustness without tamper detection) *Let D be a distribution over $\{0,1\}^{\mathsf{cr}\cdot n}\times\{0,1\}$ with a balanced "label" bit: $\Pr_{(x,y)\leftarrow D}[y=0]=1/2$. We construct a family of learning problems P_n with a family of classifiers h_n. In our construction we use a signature scheme $(\mathrm{KGen},\mathrm{Sign},\mathsf{Verify})$ for which the bit-length of vk is λ and the bit-length of signature is $\ell(\lambda)=\mathrm{polylog}(\lambda)$ and an error correction code $(\mathsf{Encode},\mathsf{Decode})$ with code rate $\mathsf{cr}=\Omega(1)$ and error rate $\mathsf{er}=\Omega(1)$.*

1. *The space of instances for P_n is $\mathcal{X}_n=\{0,1\}^{2n+n\cdot\ell(n)}$.*
2. *The set of labels is $\mathcal{Y}_n=\{0,1\}$.*
3. *The distribution D_n is defined as follows: first sample $(x,y)\leftarrow D$, then sample $(\mathsf{sk},\mathsf{vk})\leftarrow\mathrm{KGen}(1^{n\cdot\mathsf{cr}})$ and compute $[\mathsf{vk}]=\mathsf{Encode}(\mathsf{vk})$. Then compute $[x]=\mathsf{Encode}(x)$. If $y=0$ sample a random $\sigma\leftarrow\{0,1\}^{\ell(n)}$ that is not a valid signature of x w.r.t vk. Then output $(([x],\sigma^n,[\mathsf{vk}]),0)$. Otherwise compute $\sigma\leftarrow\mathrm{Sign}(\mathsf{sk},x)$ and output $(([x],\sigma^n,[\mathsf{vk}]),1)$.*
4. *The classifier $h_n\colon\mathcal{X}_n\to\mathcal{Y}_n$ is defined as*

$$h_n(x',\sigma_1,\ldots,\sigma_n,\mathsf{vk}')=\begin{cases}1 & \text{if } \exists i\in[n];\mathsf{Verify}\left(\mathsf{Decode}(\mathsf{vk}'),\mathsf{Decode}(x'),\sigma_i\right),\\ 0 & \text{otherwise.}\end{cases}$$

Proof (of Theorem 8) First it is clear that for problem P_n we have $\mathsf{Risk}_{\mathsf{P}_n}(h_n)=0$. Now we prove the computational robustness of h_n.

Claim 20 *For family P_n, and for any polynomial $s(\cdot)$ there is a negligible function negl such that for all $n\in\mathbb{N}$*

$$\mathsf{AdvRisk}_{\mathsf{P}_n,\mathsf{er}\cdot n,s}(h_n)\leq\mathrm{negl}(n).$$

Proof Similar to proof of Claim 17 we prove this based on the security of the signature scheme. Let $\mathsf{A}_{\backslash\in\mathbb{N}}$ be the family of circuits maximizing the adversarial risk for h_n for all $n\in\mathbb{N}$. We build a sequence of circuits $\mathsf{A}^1_{\backslash\in\mathbb{N}}$ and $\mathsf{A}^2_{\backslash\in\mathbb{N}}$ such that A^1_n and A^2_n are of size at most $s(n)+\mathrm{poly}(n)$. A^1_n just asks the signature for $0^{\mathsf{cr}\cdot n}$. A^2_n gets vk and does the following: It first samples $(x,y)\leftarrow D$, computes encodings $[x]=\mathsf{Encode}(x)$ and $[\mathsf{vk}]=\mathsf{Encode}(\mathsf{vk})$ and if $y=0$, it samples a random $\sigma\leftarrow\{0,1\}^{\ell(n)}$ then calls A_n on input $([x],\sigma^n,[\mathsf{vk}])$ to get $(x',(\sigma_1,\ldots,\sigma_n),\mathsf{vk}')\leftarrow\mathsf{A}_n(([x],\sigma^n,[\mathsf{vk}]),y)$.

Then it checks all σ_i's and if there is any of them that $\mathsf{Verify}(\mathsf{vk}, \sigma_i, x) = 1$ it outputs (x, σ_i), otherwise it aborts and outputs $\perp$. If $y = 0$ it aborts and outputs $\perp$. Note that A_n^2 can provide all the oracles needed to run A_n if the sampler from D, h and c are all computable by a circuit of polynomial size. Otherwise, we need to assume that our signature scheme is secure with respect to those oracles and the proof will follow. We have,

$$\begin{aligned}
&\mathsf{AdvRisk}_{\mathsf{P}_n,\mathsf{er}\cdot n,s}(h_n) \\
&= \Pr[(([x], \sigma^n, [\mathsf{vk}]), y) \leftarrow D_n;\ (x', (\sigma_1, \ldots, \sigma_n), \mathsf{vk}') \leftarrow A_n(([x], \sigma^n, [\mathsf{vk}]), y)); \\
&\quad (x', (\sigma_1, \ldots, \sigma_n), \mathsf{vk}') \in \mathsf{HD}_{\mathsf{er}\cdot n}([x], \sigma^n, [\mathsf{vk}]) \wedge h_n(x', (\sigma_1, \ldots, \sigma_n), \mathsf{vk}') \neq y].
\end{aligned}$$

Because of the error rate of the error correcting code, $(x', (\sigma_1, \ldots, \sigma_n), \mathsf{vk}') \in \mathsf{HD}_{\mathsf{er}\cdot n}(x, \sigma^n, [\mathsf{vk}])$ implies that $\mathsf{Decode}(\mathsf{vk}') = \mathsf{vk}$ and $\mathsf{Decode}(x') = x$. Also $h_n(x', (\sigma_1, \ldots, \sigma_n), \mathsf{vk}') \neq y$ implies that $y = 0$. This is because if $y = 1$, the adversary has to make all the signatures invalid which is impossible with tampering budget $\mathsf{cr} \cdot n$. Therefore y must be 1 and one of the signatures in $(\sigma_1, \ldots, \sigma_n)$ must pass the verification because the prediction of h_λ should be 1. Therefore we have

$$\begin{aligned}
\mathsf{AdvRisk}_{\mathsf{P}_n,\mathsf{er}\cdot n,s}(h_n) &\leq \Pr[((x, \sigma^n, [\mathsf{vk}]), y) \leftarrow D_n;\ (x', (\sigma_1, \ldots, \sigma_n), \mathsf{vk}') \leftarrow A((x, \sigma, [\mathsf{vk}]), y)); \\
&\qquad y = 0 \wedge \exists i \mathsf{Verify}(\mathsf{vk}, \sigma_i, x)] \\
&\leq \Pr[(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathrm{KGen}(1^n);\ 0^{\mathsf{cr}\cdot n} \leftarrow \mathsf{A}_1(1^n, \mathsf{vk});\ \sigma \leftarrow \mathrm{Sign}(\mathsf{sk}, 0^{\mathsf{cr}\cdot n}); \\
&\qquad (x, \sigma_i) \leftarrow \mathsf{A}_2(\mathsf{vk});\ \mathsf{Verify}(\mathsf{vk}, x, \sigma_i)]
\end{aligned}$$

Thus, by the unforgeability of the one-time signature scheme we have

$$\mathsf{AdvRisk}_{\mathsf{P}_n,\mathsf{er}\cdot n,s}(h_n) \leq \mathrm{negl}(n).$$

■

Now we show that h_n is not robust against computationally unbounded attacks.

Claim 21 *For family* P_n *and any* $n \in \mathbb{N}$ *we have*

$$\mathsf{AdvRisk}_{\mathsf{P}_n,\ell(n)}(h_n) = 0.5.$$

Proof For any $(([x], \sigma^n, [\mathsf{vk}]), y)$ define $A([x], \sigma^n, [\mathsf{vk}])$ as follows: If $y = 1$, A does nothing and outputs $([x], \sigma^n, [\mathsf{vk}])$. If $y = 0$, A search all possible signatures to find a signature σ' such that $\mathsf{Verify}(\mathsf{vk}, \sigma', x) = 1$. It then outputs $([x], (\sigma', \sigma^{n-1}), [\mathsf{vk}])$. Based on the fact that the size of signature is $\ell(n)$, we have $\mathsf{HD}((x, (\sigma', \sigma^{n-1}), [\mathsf{vk}]), (x, \sigma^n, [\mathsf{vk}])) \leq \ell(n)$. Also, it is clear that $h_n(x, (\sigma', \sigma^{n-1}), [\mathsf{vk}]) = 1$ because the first signature is always a valid signature. Therefore we have

$$\begin{aligned}
\mathsf{AdvRisk}_{\mathsf{P}_n,\ell(n)}(h_n) &\geq \Pr[(([x], \sigma^n, [\mathsf{vk}]), y) \leftarrow D_n; h(A(([x], \sigma^n, [\mathsf{vk}]))) \neq y] \\
&= \Pr[(([x], \sigma^n, [\mathsf{vk}]), y) \leftarrow D_n; 1 \neq y] \\
&= 0.5.
\end{aligned}$$

■

This concludes the proof of Theorem 8. ■

Appendix C. Proof of Theorem 10

Before proving Theorem 10, we recall a useful lemma. The same proof of amplification of (weak to strong) one-way functions by Yao (1982) and described in (Goldreich, 2007), or the parallel repetition of verifiable puzzles (Canetti et al., 2005; Holenstein and Schoenebeck, 2011) can be used to prove the following lemma.

Lemma 22 (Amplification of verifiable puzzles) *Suppose S is a distribution over Boolean formulas such that for every poly-size adversary A, for sufficiently large n, it holds that solving the puzzles generated by S are weakly hard. Namely, $\Pr_{\phi \leftarrow S(1^n, r_1)}[\phi(t) = 1; t \leftarrow \mathsf{A}(\phi)] \leq \varepsilon$ for $\varepsilon = 1/\mathrm{poly}(n)$. Then, for any polynomial-size adversary A, there is a negligible function negl, such that the probability that A can simultaneously solve all of $k = n/\varepsilon$ puzzles $\phi_1, \ldots, \phi_k$ that are independently sampled from S is at most $\mathrm{negl}(n)$.*

Proof (of Theorem 10.) First consider the following sampler S_1. (We will modify S_1 later on).

1. Sample m examples $\mathcal{S} \leftarrow D_n^m$.
2. Run L to get $h \leftarrow L(\mathcal{S})$.
3. Sample another $(x, y) \leftarrow D_n$
4. Using the Cook-Levin reduction, get a Boolean formula $\phi = \phi_{h,x,y}$ such that $\phi \in \mathrm{SAT}$, if (1) $\mathbf{d}(x', x') \leq b(n)$ and (2) $h(x') \neq y$. This is possible because using h, x, y, both conditions (1) and (2) are efficiently checkable.
5. Output ϕ.

By the assumptions of Theorem 10, it holds that $\Pr_{\phi \leftarrow S_1}[\phi \in \mathrm{SAT}] \geq \beta(n)$ while for any poly-size algorithm A, it holds that $\Pr_{\phi \leftarrow S_1, t \leftarrow \mathsf{A}(\phi)}[\phi(t) = 1] \leq \alpha$. So, S_1 almost gets the conditions of a hard sampler for $\mathbf{NP}$, but only with a weak sense.

Using standard techniques, we can amplify the ε-gap between $\alpha, \beta(n)$. The algorithm S_2 works as follows. (This algorithm assumes the functions $\alpha, \beta(n)$ are efficiently computable, or at least there is an efficiently computable threshold $\tau \in [\alpha + 1/\mathrm{poly}(n), \beta(n) - 1/\mathrm{poly}(n)]$.)

1. For $k = n/\varepsilon^2$, and all $i \in [k]$, get $\phi_i \leftarrow S_1$.
2. Using the Cook-Levin reduction get a Boolean formula $\phi = \phi_{\phi_1, \ldots, \phi_k}$ such $\phi \in \mathrm{SAT}$, if there is a solution to satisfy at least $\tau = (\alpha + \beta(n))/2$ of the formulas $\phi_1, \ldots, \phi_k$. More formally, $\phi \in \mathrm{SAT}$, if there is a vector $\bar{t} = (t_1, \ldots, t_k)$ such that $|\{i \colon \phi_i(t_i) = 1\}| \geq \tau$. This is possible since verifying $\bar{t}$ is efficiently possible.

By the Chernoff-Hoeffding bound,

$$\Pr_{\phi \leftarrow S_2}[\phi \in \mathrm{SAT}] \geq 1 - e^{-(\varepsilon/2)^2 \cdot n/\varepsilon^2} \geq 1 - e^{-n/4}.$$

Proving the second property of the hard sampler S is less trivial, as it needs an efficient *reduction*. However, we can apply a weak bound here and then use Lemma 22. We first claim that for any poly-size adversary A,

$$\Pr_{\phi \leftarrow S_2, t \leftarrow \mathsf{A}(\phi)}[\phi(t) = 1] \leq 1 - \varepsilon/3. \tag{1}$$

To prove Equation 1, suppose for sake of contradiction that there is such adversary A. We can use A and solve $\phi' \leftarrow S_1$ with probability more than $\alpha + \Omega(\varepsilon)$ which is a contradiction. Given ϕ', The reduction is as follows.

1. Choose $i \leftarrow [k]$ at random.
2. Sample $k-1$ instances $\phi_1, \ldots, \phi_{i-1}, \phi_{i+1}, \ldots, \phi_k \leftarrow S_1$ independently at random.
3. Let $\phi_i = \phi'$.
4. Ask A to solve $\phi_{\phi_1,\ldots,\phi_k}$, and if A's answer gave a solution for $\phi_i = \phi'$, output this solution.

Since A cannot guess i, a simple argument shows that the above reduction succeeds with probability $\alpha + \varepsilon/2 - \varepsilon/3 = \alpha + \varepsilon/6$. Now that we have a puzzle generator S_2 that has satisfiable puzzles with probability $1 - \mathrm{negl}(n)$ and efficient algorithms can solve its solutions by probability at most $\varepsilon/2$, using Lemma 22, we can use another direct product and design sampler S that samples $2n/\varepsilon$ independent instances from S_2 and asks for solutions to all of them. Because we already established that $\Pr_{\phi \leftarrow S_2}[\phi \in \mathrm{SAT}] \geq 1 - \mathrm{negl}(n)$, the puzzles sampled by S are also satisfiable by probability $1 - n \cdot \mathrm{negl}(n) = 1 - \mathrm{negl}(n)$, but efficient algorithms can still find the solution only with probability that is $\mathrm{negl}(n)$. ■